UC Davis
IDAV Publications

Title
Scientific Visualization Techniques for Volume Fraction Data and Function Fields

Permalink
https://escholarship.org/uc/item/41x9x1gm

Author
Anderson, John C.

Publication Date
2009

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Scientific Visualization Techniques for
Volume Fraction Data and Function Fields

By

JOHN CARL ANDERSON
B.A. History (University of the Pacific) 2004
B.S. Computer Science (University of the Pacific) 2004
M.S. Computer Science (University of California, Davis) 2008

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Prof. Kenneth I. Joy

Prof. Nina Amenta

Dr. Mark A. Duchaineau

Committee in charge

2009

© 2009 John Carl Anderson

Acknowledgments

This dissertation represents many years of work: five years of my own effort; that and much more from my advisor Ken Joy; work by Mark Duchaineau and others at Lawrence Livermore National Laboratory in mentoring and technical collaboration; the work of my co-authors; the work of Jamie Jones, Patricia Gomez, and Zhi-Wei Lu to make the Institute for Data Analysis and Visualization at UC Davis run like a well-oiled machine; work by the administrative staff of the University Relations Program and Institute for Scientific Computing Research at LLNL to help students; work by those whose data I visualized, and whose funding I received; work by my fellow students to make IDAV a great place to spend long hours; constant support from my wife Chantel and both our families; and the work of countless others.

I feel privileged to have had the opportunity to pursue this research for the past few years, and I am forever indebted to all who have worked so hard to make that possible.

Thank you.

Contents

List of Figures

List of Tables

John Carl Anderson
June 2009
Computer Science

Scientific Visualization Techniques for

Volume Fraction Data and Function Fields

Abstract

While the scientific visualization community is comfortable with isosurfacing and volume rendering of scalar fields, data from simulations and sensors often have additional constraints or dimensions that are not easily handled by these algorithms. In the first part of this dissertation we consider volume fraction data and the material interface reconstruction problem, for which existing isosurfacing and segmentation methods do not produce satisfactory results. Optimization-based methods are introduced that produce accurate multi-material segmenting surfaces through volume fraction data. In the second part, we discuss visualization techniques for function fields. A dimension reduction approach based upon probing and range-space segmentation is introduced, allowing function fields to be analyzed with traditional visualization algorithms. Finally, queries are considered for explicit feature extraction.

Chapter 1

Introduction

The majority of the research in scientific visualization has concentrated on scalar fields, where points in n-dimensional space are mapped to scalar values:

$$F : p \in \mathbb{R}^n \rightarrow s \in I,$$

where I is a closed interval in $\mathbb{R}$. Several mature visualization techniques exist for scalar fields, including isosurfaces [48], slicing, and volume rendering [12]. Most often, these methods have focused on static or time-varying 2- and 3-dimensional spatial domains.

The size and complexity of scientific datasets continues to grow, however, with increasing computing power and our ability to gather more and more data via increasingly powerful imaging and sensor technology. Scalar fields alone are insufficient in many scientific domains; datasets that represent physical phenomena now contain billions of multi-valued, multi-dimensional, time-varying elements, and we are no longer able to fully analyze them.

This work focuses upon two data types – volume of fluid data and function fields – which, by constraint or dimension, are not easily handled by traditional visualization algorithms. In the first part of this dissertation we consider surfaces, and the challenges presented by volume of fluid data. In the second, we focus upon visualization techniques for function fields.

1.1 Volume Fraction Data

In the first part of this dissertation, we focus on volume fraction data and the material interface reconstruction problem, for which existing isosurface and segmentation methods do not produce satisfactory results. Consider a spatial domain which has been decomposed into a set of cells C – e.g., a rectilinear grid. In an n-material problem, each cell $c \in C$ has an associated tuple $(m_1, \ldots, m_n)$; the value m_i is the fractional volume of material i within the cell. The fractions within each cell's tuple are positive, and account for the entire volume of the cell. The most common task for volume fraction data is Material Interface Reconstruction (MIR). The goal of MIR is to construct boundary interfaces between material regions such that within each cell the fractional volumes of each material matches the given volume fractions. Optimization-based methods are introduced in Chapters 3 and 4 that produce accurate multi-material segmenting surfaces through volume fraction data in two and three dimensions.

1.2 Function Fields

In the second part of this dissertation, we discuss visualization and analysis techniques for function field data. Function fields directly arise in applications where an entire spectrum of values is simulated or collected at each data point. From hyperspectral imagery to ground cover distributions, ocean, weather, and air quality simulations, we find data in which samples do not correspond to collections of disjoint scalar values, but rather one-dimensional scalar functions:

$$F : p \in \mathbb{R}^n \rightarrow f_p \in \mathscr{F}_I,$$

where $\mathscr{F}_I$ is the set of functions over a closed interval I. Simply, they can be thought of as large, vector-valued datasets, where the functions are sampled as m-dimensional vectors.

Function fields present challenges for traditional visualization techniques such as isosurfacing and volume rendering due to the addition of a functional domain. In Chapter 5, we further describe

function fields and the challenge their extra dimensionality presents in visualization. We introduce a new dimension reduction approach based upon probing and range-space segmentation in Chapter 6, which allows function fields to be analyzed with traditional visualization algorithms. In Chapter 7 we describe feature extraction in function fields using queries.

Part I

Volume Fraction Data

Chapter 2

The Material Interface Reconstruction Problem

Interface reconstruction and tracking continues to be an important problem with broad application. Scientific simulation of fluid transport naturally produces interfaces – the boundaries between sloshing fluids in tanks, between cavity and casting in mold filling applications, and waves breaking on shorelines. Interface problems arise in combustion applications, climate studies, astrophysics, and medical imaging.

Isosurfacing is one of the oldest, and best understood interface reconstruction problems in scientific visualization. Here the task is to produce the segmenting surface between two regions of a mesh-based dataset: regions with scalar values less than the isovalue ($-$), and regions with scalar value greater than the isovalue ($+$). The seminal work on isosurface extraction is Marching Cubes (MC) by Lorensen and Cline [48]. Since the late 1980s, isosurfacing has been extensively studied, generalized, and broadened by the scientific visualization community. An important development has been the extension from binary labelings ($-/+$) to multiple labelings (e.g., A, B, C, etc.); interfaces for problems containing three or more material labels can be extracted with multi-label Marching Cubes methods [31, 84, 7], Dual Contouring [38, 67], or the tetrahedral method of Nielson and Franke [57].

In many applications, however, it is necessary to reconstruct interfaces between multiple material regions of *known volume* rather than a known labeling. Multi-fluid hydrodynamics simulation codes,

for example, often produce datasets in which cell-centered scalar quantities report the fractional volume of each material contained within each cell; volume fractions are also found in studies of partial differential equations, fluid applications, probability calculations, and neutron transport. The challenge then is to create high-quality reconstructions – i.e., boundary interfaces between different material regions – for visualization and analysis to help study the results from such a simulation.

Volume fraction data exists within a spatial domain that has been decomposed into a finite grid of cells C. In an n-material setting, each cell $c \in C$ has an associated tuple $V_c = (v_1, \ldots, v_n)$, where the value v_i is the fractional volume of material i within the cell. Volume fractions are non-negative ($v_i \geq 0$), and form a partition of unity over the volume of the cell ($\sum_{i=1}^{n} v_i = 1$). *Mixed* cells have multiple non-zero volume fractions; *pure* cells contain only one material. *The material interface reconstruction (MIR) problem is to construct boundary interfaces between regions of homogeneous material, while accurately approximating per-cell volume fractions.*

After some study, one realizes that the material interface reconstruction problem is fundamentally different from the classic isosurface and multi-material segmentation problem often studied by the visualization community. Not only must we generate smooth, continuous segmenting surfaces that partition cells into homogeneous material regions, but those surfaces must be embedded within the spatial domain so as to reproduce the volume fractions within each cell.

Most work on MIR has been driven by computational fluid dynamics (CFD) research. CFD codes often prefer to use a combination of Lagrangian and Eulerian formulations [8]. In the Lagrangian paradigm, the underlying mesh of the simulation domain is moved and potentially modified over time. When simulating complicated flow, however, it is often simpler to move to an Eulerian framework in which fluids are advected from cell to cell over a static mesh. Arbitrary Lagrangian-Eulerian (ALE) codes combine these two schemes – i.e., advection of material over a flexible mesh – for the accurate simulation of complex flows.

The Volume-of-Fluid (VOF) method is often used during the simulation of Eulerian multi-fluid hydrodynamic flows [33]. In a VOF simulation, fractional material volumes are maintained for each cell. To advance the simulation, interface geometry is reconstructed in order to calculate the

flux of material between cells. Storing per-cell volumes, rather than explicit boundary interface geometry, eases the simulation of complicated flows, however the reconstruction material boundary interface is a crucial part of accurately advecting materials [36].

Reconstruction methods can be broadly split into simulation and visualization approaches. The reconstruction methods employed within VOF simulation are a crucial part of accurately advecting materials [36], and understandably the focus is upon volume conservation and convergence. In the visualization setting, the goal is to produce high-quality surface meshes that approximate the volume fractions with low error and integrate into the visualization and analysis framework of existing tools (e.g., [15]).

One of the first simulation methods is Simple Line Interface Calculation (SLIC), described by Noh and Woodward [59], where cells are simply partitioned with axis-aligned planes, such that the total material volume in each cell is correct. The Piecewise Linear Interface Calculation (PLIC) algorithm of Youngs [86] is similar to SLIC, however each cell is partitioned by planes aligned to local material "gradients." While PLIC is fast and preserves volume fractions, the reconstruction is discontinuous across cell boundaries and thus not suitable for visualization. Furthermore, PLIC is ambiguous for three or more materials due to the ordering of its binary segmentations.

Pilliod and Pucket [36] describe two modifications to the PLIC method, both of which use least-squares to minimize the error of approximately linear interfaces. Garimella et al. [27] demonstrate how to fix certain local topological inconsistencies in PLIC reconstructions. Dyadechko and Shashkov [20, 21, 22] and Schofield et al. [68] present interface reconstruction algorithms for volume fraction data augmented with material centroid information. These methods either inherit the partitioning ambiguity of PLIC, or produce discontinuous, piecewise linear boundaries.

Several visualization methods have attempted to recast this problem into an isosurface-like problem, but these methods compromise the data, create artifacts, have a large error, or require assumptions that cause incorrect reconstructions. Bonnell et al. [10, 9] move the calculations to the dual grid, where each vertex holds the volume fraction of an original cell. They calculate intersections using barycentric interpolation in the space of the volume fractions. One problem with this approach

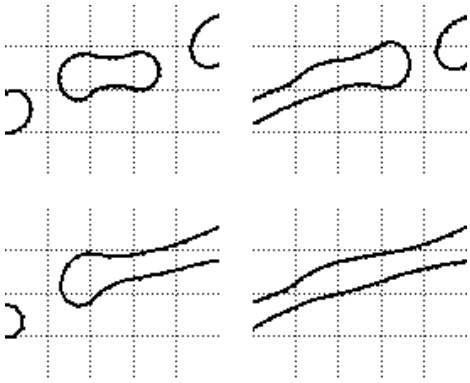


Figure 2.1: Four reconstructions for the same volume fraction data.

is that interfaces are not calculated from the original data, but from a dual grid, which induces significant error. Meredith [54] averages volume fractions to mesh vertices, and then utilizes an isosurface approach. Both methods, however, miss small scale features entirely (e.g., filaments and thin shells), have problems with many materials, and do not preserve volume fractions.

2.1 Goals and Contributions

It should be noted that this is an under-constrained problem, since there are an infinite number of cell partitions that can satisfy a given set of volume fractions (consider Figure 2.1). Table 2.1 outlines some goals and metrics that may be used as a guide to evaluate reconstruction methods. In volume of fluid simulations, for example, the convergence properties of a reconstruction is important as grid resolutions increase: Youngs' method is first-order accurate [86], while Pilliod and Pucket describe second-order methods [36]. Material ordering and boundary continuity are also important in simulation methods [27, 20, 21, 22], especially when centroid data is present [68]. For visualization methods, various aspects of surface quality have been highly valued. The method of Meredith [54], for example, strives to reduce the geometry produced by the method of Bonnell et al. [10, 9], but does little to improve volume accuracy.

In this dissertation, we attempt to fill a gap in the literature of MIR methods. Specifically, our goal is to produce interfaces that provide a simple explanation of the volume fractions (in terms of topology), are geometrically smooth, continuous across cell boundaries, and segment cells into

Goal	**Metric**
Performance	*Time*
	# Primitives
Surface Quality	*Volume Preservation*
	Continuity
	Area
	# Connected Components
	# Primitives
Convergence	*First-order, second-order, ...*

Table 2.1: While MIR is under-constrained, certain goals and metrics may be used as a guide to evaluate reconstruction methods.

regions that approximate the problem's volume fractions with low error. Generating high-quality, volume-accurate material interfaces allows for meaningful calculations of interface surface statistics such as area and curvature, allows us to texture accurately the interface mesh with other data items from the simulations (temperature, pressure, etc.), and dramatically increases the utility of material interface visualizations.

To this end, we introduce two optimization-based methods, each of which is able to handle more than two materials while producing interfaces with low volume error. Chapter 3 develops a discretized approach to MIR based upon labeling volume elements through an energy minimization process. Chapter 4 builds upon this result by developing a topology generation pipeline for volume fraction data, and introduces a dynamic surface model to produce smooth, volume-accurate material interfaces.

Chapter 3

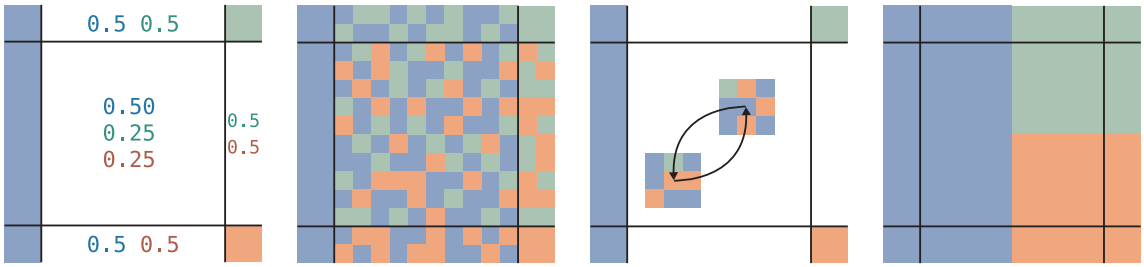
Discrete Multi-Material Interface Reconstruction

In this chapter, we reinterpret MIR as a segmentation problem over a discretization of the problem's original spatial domain. Our formulation eases the extraction and visualization of material interfaces, and unlike previous work:

- material volume is preserved with bounded error,
- interfaces are continuous across cell boundaries,
- interfaces have low surface area and curvature, and
- reconstruction works for time-varying and static data of arbitrary dimensionality.

Additionally, our technique scales well with respect to material interface complexity, and is easily parallelized.

The basis of our approach is to discretize cells containing more than one material into small, fractional volume elements. Each of these "subcells" is then labeled as being entirely one material or another based upon the problem's volume fractions. Producing a good labeling of the subcells is a non-trivial problem, however an initial labeling can be effectively optimized. In our work, each subcell is attributed a simple, local energy equal to the number of its neighboring subcells with a



(a) Problem (b) Initial State (c) Swap (d) Converged State

Figure 3.1: Algorithm overview focusing upon a single mixed cell with both pure and mixed neighbors (a). During optimization of the labeled discretization (b) a *volume conservative swap* of two subcell labels (c) is performed probabilistically, based upon its effect on the Potts-model energy. The converged material interface reconstruction produced by our method is shown in (d).

different label. Known as the Potts-model energy (see [83]), this metric has been widely used in interface problems, from studying cellular structures [30] to interpolating region boundaries between segmented images [19]. Optimizing the Potts-model energy over the discretization leads to a labeling with low surface area and curvature – desirable properties that translate to our final interface reconstruction.

Working in a discretized setting greatly simplifies the construction of material interfaces. In our method, material interfaces are surfaces that separate regions of the discretization with different material labels. Surface mesh representations of these interfaces can be easily extracted, even in cases with complex topology such as multi-material junctions and multiple intersections along a single mesh edge.

The next section details how we cast interface reconstruction as an optimization problem over a discrete, labeled grid. Techniques for extracting and visualizing material interfaces from the discretization are detailed in Section 3.2, followed by some notes on the implementation of our method in Section 3.3. Finally, we present results of our work over two- and three-dimensional fluid flow datasets in Section 3.4.

3.1 Method

Consider a spatial domain that has been decomposed into a finite grid of cells C. In an n-material problem, each cell $c \in C$ has an associated tuple $V_c = (v_1, \ldots, v_n)$, where the value v_i is the fractional volume of material i within the cell. Volume fractions are non-negative ($v_i \geq 0$), and account for the entire volume of the cell ($\sum_{i=1}^{n} v_i = 1$). *Pure* cells are entirely one material, while *mixed* cells have multiple non-zero volume fractions. Figure 3.1(a) shows a hypothetical MIR problem in which pure cells are shown in a solid color.

Our method begins with a discretization step. Each cell is subdivided into S subcells of uniform fractional volume $dA = \frac{1}{S}$ to form the discretization D. We allow each subcell, in turn, to be assigned a label corresponding to one of the n materials.

In this discrete setting, we formulate material interfaces as separating surfaces between regions of D with different material labels. After discretization, therefore, the goal becomes to generate a simple labeling of the subcells such that problem's volume fractions are preserved as closely as possible. Our approach – described in the remainder of this section – is to first produce an initial, valid labeling and then apply optimization.

We begin by randomly assigning material labels to subcells with the constraint that for each label $i \in (1, \ldots, n)$ there are approximately $\frac{v_i}{dA}$ subcells with label i in cell c. Figure 3.1(b) illustrates the initial state of D after labels have been assigned based upon the volume fractions shown in 3.1(a).

To improve the labeling we define a local measure of the labeling quality. In this chapter, we use a discrete estimate of the labeling smoothness known as the Potts model [83]. Consider a labeling f of D such that f_x is the label of subcell x. The Potts-model energy at x is the number of subcells neighboring x with a different label:

$$E_x(f) = \sum_{y \in N} W_{x,y} \cdot \delta(f_x \neq f_y), \tag{3.1}$$

where W is a weighting function for offsets within the local neighborhood N (which may span

original cell boundaries), and $\delta = \{\text{true} : 1; \text{false} : 0\}$.

Extending the Potts-model energy over the entire discretization

$$E(f) = \sum_{x \in D} E_x(f), \tag{3.2}$$

allows for optimization of the labeling through energy minimization. The end result of optimization will be a smoother, simpler labeling and improved material interfaces.

We optimize the energy function in Equation 3.2 using simulated annealing [43] in order to have explicit control of how the labeling is changed. More recent techniques such as graph cuts [11, 45] are not used because their optimization moves do not conserve volume, a firm requirement in our application.

In simulated annealing, changing from one state to another – i.e., from a labeling f to a new labeling f' – is allowed probabilistically as a function of the annealing temperature T and the corresponding change in energy ΔE:

$$P = \begin{cases} 1 & \Delta E < 0, \\ e^{-\Delta E/T} & \text{otherwise.} \end{cases} \tag{3.3}$$

Changes that improve the labeling are always taken. Changes that increase the total energy remain likely when T is high, however as the temperature decreases, the system converges because those changes become much less likely.

Per-cell volume can be maintained by restricting the labeling changes considered during optimization. In our approach, we only allow the *volume conservative swap* of two labels. Here, the labels of two randomly chosen subcells – x and y – within a cell c are exchanged to produce a new labeling, as shown in Figure 3.1(c). Consider the initial Potts-model energy of the subcell pair (x, y) under the labeling f:

$$E_{x,y}(f) = E_x(f) + E_y(f).$$

Exchanging the labels of this pair would produce a new labeling f' with energy $E_{x,y}(f')$, in which the total per-cell, per-material volumes remain unchanged. The change in energy $\Delta E = E_{x,y}(f') - E_{x,y}(f)$ produced by the swap can be used to determine if the labeling f' is accepted during optimization (Equation 3.3).

Using volume conservative swaps guarantees that the labeling f accurately reflects the problem's volume fractions throughout the optimization process. Thus, an upper bound on the per-cell error $\varepsilon(c)$ of the discretization labeling is:

$$\varepsilon(c) \leq \begin{cases} (n-1)dA & \text{if } c \text{ is mixed,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

This error bound is driven by the subdivision rate S, and the resulting quantization of the cell's volume fractions into multiples of size dA. Optimization may be performed for an arbitrarily short or long period of time depending on the quality of the labeling desired. Sections 3.3 and 3.4 provide insight into the convergence behavior of our approach.

3.2 Visualization

The labeling of fractional volume elements as entirely one material or another – described in the previous section – explicitly encodes the characteristic function of our MIR solution. In this section, we discuss methods for visualizing material regions and constructing surface mesh representations of material interfaces in this discrete context.

Material regions can be directly visualized in our approach. Generally speaking, we assign a unique color to each material and render the discretization D colored by its current labeling f. In 2D this produces a color image, and in three dimensions results in an image cube which can be visualized using volume rendering. Time-varying volume fraction data naturally leads to a sequence of multiple images. Visualizing material regions is attractive in 2D since occlusion is not an issue; correspondingly, in this chapter we render material regions rather than interfaces for all two-dimensional

datasets.

Material interfaces are also simple to extract: interfaces in our discrete formulation are surfaces that separate regions of D with different material labels. A surface mesh representation of material interfaces can be constructed by extracting co-incident faces between adjacent subcells with different labels.

Surface meshes constructed in this way are able to capture simple and complex interface topologies, such as multi-material junctions and multiple intersections along a single mesh edge. They also exactly match the volume fractions given by the labeling f, and the problem's volume fractions with bounded error (Equation 3.4). Upon close inspection, however, boundaries constructed in this manner can be unpleasant to visualize because they capture sharp boundaries at the sub-cell scale.

An alternate surface construction option is to apply a multi-material segmentation algorithm over an approximate, smoothed version of the labeling field. For two-material problems interfaces can be extracted using Marching Cubes [48]. Interfaces in problems with three or more materials can be extracted using one of various multi-label segmentation algorithms, such as multi-label Marching Cubes methods [31, 84, 7], Dual Contouring [39], or the method of Nielson and Franke [57] on an implicit tetrahedrization of the rectilinear domain. We have found that filtering f with a narrow Gaussian kernel improves material interfaces for visualization without introducing large error (see Section 3.5 for a discussion of the effects of smoothing upon volume preservation).

3.3 Implementation

In this section, we provide some implementation details regarding topics such as performance, convergence, neighborhood size and weighting, and accuracy.

Performance There are two important areas of performance to consider: computation and memory consumption. In terms of computation, simulated annealing optimization of the labeling energy is not cheap. However, it is straightforward to develop a highly parallel implementation

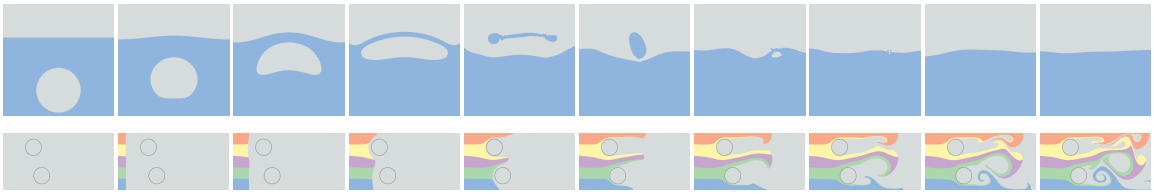


Figure 3.2: Evolution in time of the two-dimensional fluid flows considered in this chapter; interfaces were reconstructed by our method. On the top, a bubble of low density fluid rises through a denser fluid. On the bottom, five fluids pass two cylinders (six total materials).

over independent sets of cells using "checkerboarding" [82]. Interactive visualization also hides the cost of optimization.

The memory requirement per subcell without encoding is a single byte for $n < 256$ materials. Furthermore, memory usage is reduced by the sparseness of mixed material cells; often the vast majority of cells in a volume fraction dataset do not contain interfaces. It is efficient to only subdivide mixed material cells, thus allowing our method to scale with complexity of the material interfaces rather than the size of the problem domain.

Convergence In some applications, the quality of simulated annealing *can* be strongly influenced by the *annealing schedule* – i.e., how the temperature T changes over time. We have found, however, that the annealing schedule is not a crucial factor in our method. This is because the entire system starts very close to a local minimum before optimization: pure cells heavily influence neighboring mixed cells, but do not change themselves. Setting the temperature to a low constant allows the system to consistently converge to a reasonable reconstruction without a complicated annealing schedule. For all results presented in this chapter we have used $T = 0.25$.

Neighborhood The neighborhood N and weight function W used in Equation 3.1 are also important. In this chapter we consider the neighbors of subcell x to be its directly incident subcells – i.e., 8 neighbors in two dimensions, and 26 neighbors in three dimensions, etc. For time-varying data, the neighborhood can also be extended over time to encourage temporal coherence. The weight between two subcells is simply the inverse magnitude of the offset between the two subcells.

Accuracy The accuracy of our reconstruction in terms of volume conservation is determined by

the level of subdivision used for discretization (Equation 3.4). Higher levels of subdivision lower the error bound, however convergence will take longer. For d-dimensional rectilinear grids we can define the subdivision rate R, such that $S = R^d$. In practice, we have found that subdivision rates between $R = 5$ and $R = 10$ produce good results with fast convergence in two and three dimensions. The upper bound on error for a 2 material problem in three-dimensions is 0.8% with $R = 5$, and 0.1% with $R = 10$ (125 and 1000 subcells per cell, respectively). Note that these bounds apply to non-smoothed interfaces; in Section 3.5 we discuss the empirical error of smoothed surfaces.

3.4 Results

We have tested our method across multiple volume fraction datasets resulting from CFD simulations in two- and three-dimensions. Results in this section were obtained with a multi-threaded software implementation on an Apple MacBook Pro notebook computer (2.33 GHz Intel Core 2 Duo processor, 2 GB memory, and an ATI Radeon X1600 graphics card).

Our first dataset was generated from a two-dimensional simulation of a low density fluid bubble rising through a denser fluid. The computational domain was a 64^2 rectilinear grid. The top row of Figure 3.2 provides an overview of this flow, reconstructed by our method, as it evolves over 200 timesteps. The top row of Figure 3.3 compares our interface reconstruction to PLIC over a 13x10 cell window; our reconstruction produces simpler, smoother interfaces while preserving volume from the original data to within 1% error. Subdivision was set to 10^2 subcells per mixed cell, and simulated annealing was performed for 10 seconds per timestep prior to visualizing the material interfaces.

The next dataset is from a two-dimensional simulation of five fluids passing two cylinders. The computational domain was 128x64. Our reconstructions of this flow use a 10^2 subcell per cell subdivision. The bottom row of Figure 3.2 provides an overview of this flow, reconstructed by our method, as it evolves over 256 timesteps. Due to the method of simulation, the cylinders and "empty" space (in grey) are modeled as a sixth material. While we show geometry of the cylinders

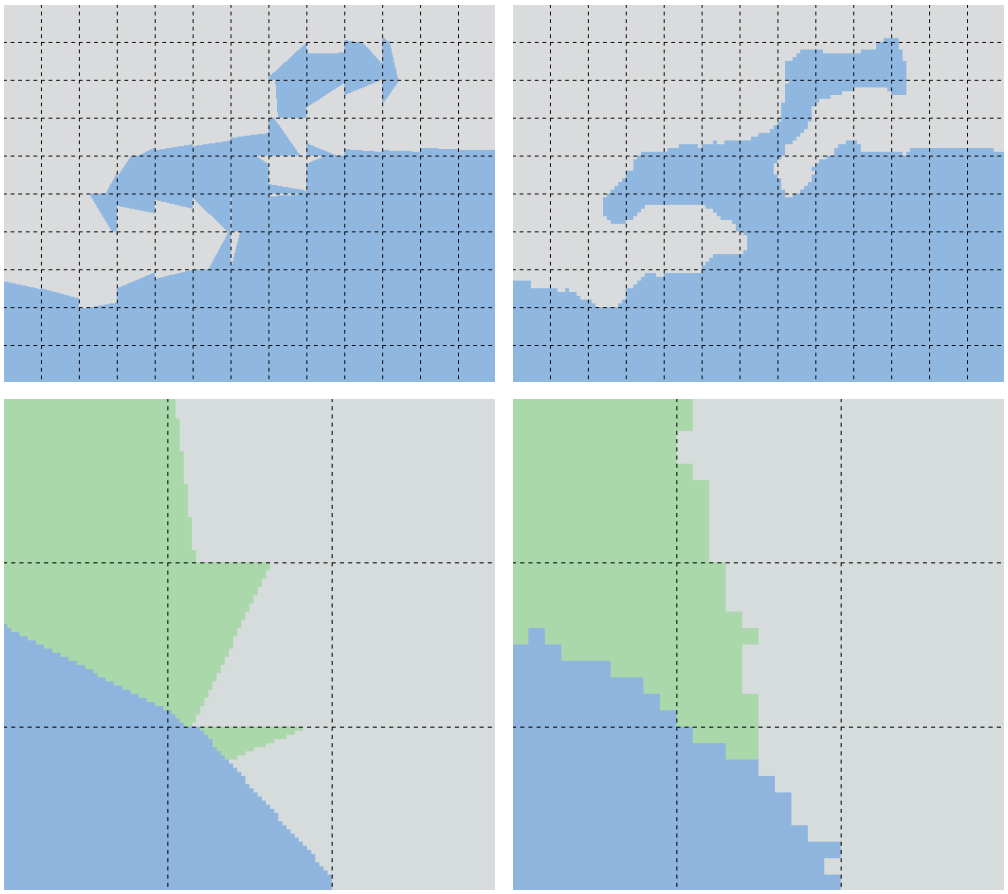


Figure 3.3: Close-ups of regions reconstructed by PLIC (left) and our method (right). Images in the top row are from the two-dimensional bubble dataset; the bottom row shows a "T-junction" between three materials in the two cylinders dataset.

for clarity, the geometry is neither part of volume fraction dataset, nor known to our MIR algorithm. In the bottom row of Figure 3.3 we show a close-up of a 3x3 cell window in which a "T-junction" between three materials is located; our method, while discretized, better captures the behavior of the interfaces around the junction. We also use this flow to illustrate the convergence of our method. Figure 3.4 shows a single timestep of this flow with approximately 7% mixed cells. Simulated annealing was performed on different parts of the discretization labeling for different lengths of time: the top third was left in the initial state without optimization, the middle third was optimized for 1 second, and the bottom third was optimized for 10 seconds. While in the most complex cases, the upper error bound remains 2% due to discretization regardless of optimization, the interfaces become simpler and smoother with brief optimization. Simulated annealing for longer than 10 seconds per timestep does not significantly improve the results.

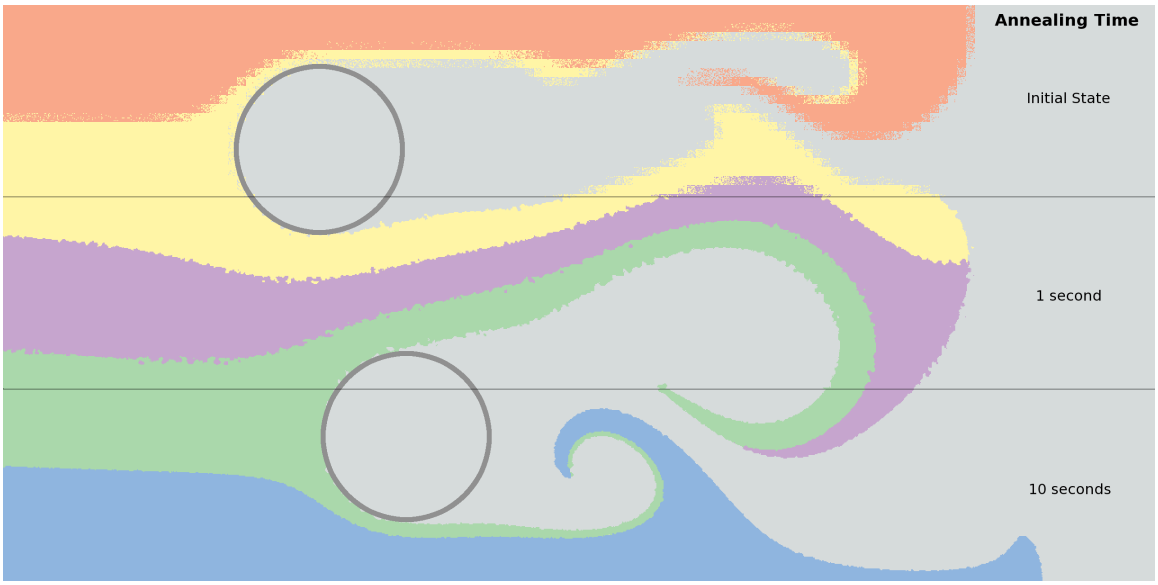


Figure 3.4: Simulated annealing was performed on different parts of the discretization labeling for different lengths of time: the top third was left in the initial state without optimization, the middle third was optimized for 1 second, and the bottom third was optimized for 10 seconds.

Our final dataset is an extension of the bubble simulation to three dimensions: again, a low density fluid bubble rises through a denser fluid. The grid is now 64^3, and the simulation consists of 100 timesteps. Figure 3.5 provides a split view of interfaces reconstructed from this dataset as the bubble bursts: on the left interfaces were extracted by isosurfacing the smoothed discretization labeling (as described in Section 3.2), and on the right "surfaces" were generated by PLIC. Subdivision was set to 5^3 subcells per cell, and simulated annealing was performed for 10 seconds per timestep prior to visualization of the material interfaces.

Section 3.3 noted that the sparseness of cells containing material interfaces leads to memory savings when subdivision is only performed over mixed cells. Figure 3.6 plots the percentage of mixed cells over time for each fluid flow dataset considered in this chapter. For both bubble datasets, the percentage is very low – below 6% – over all timesteps. For the five fluids passing two cylinders dataset, the percentage of mixed cells increases over time because the material interfaces become more complex. However, even for the most complex timestep, the storage cost of the discretization is reduced approximately 90% by only subdividing mixed cells.

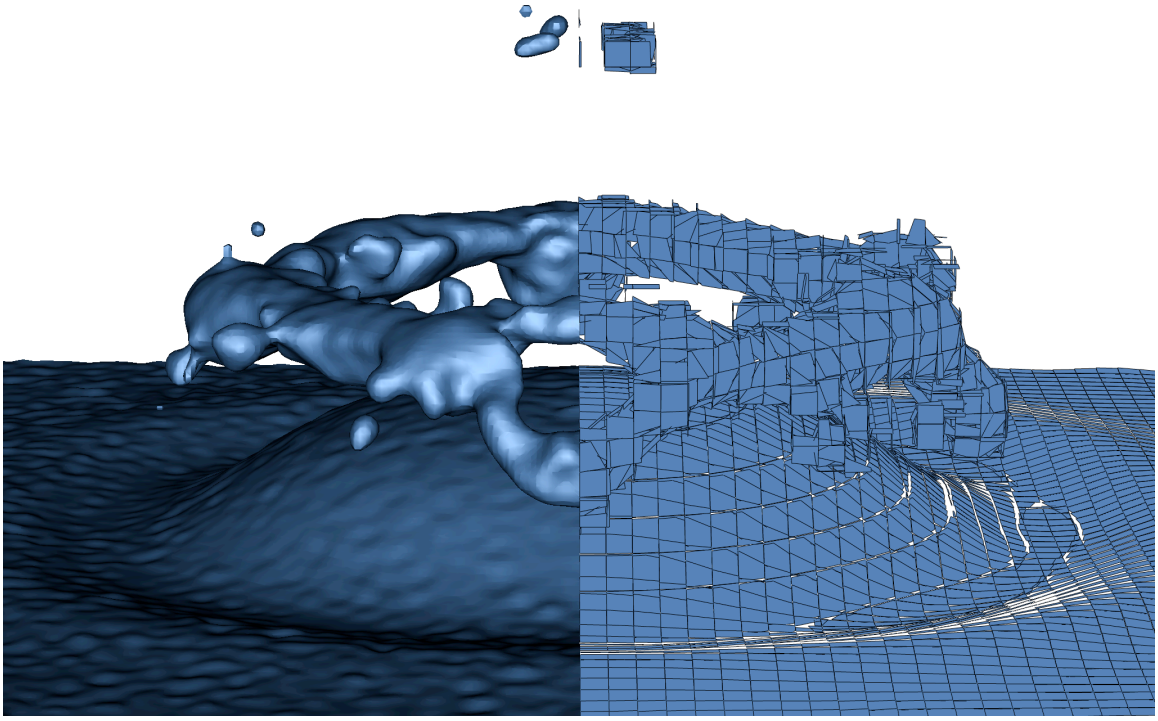


Figure 3.5: Interface reconstructions of the three-dimensional bubble dataset as the bubble bursts. On the left, we show the interface extracted from our discretization labeling; on the right is the discontinuous 3D PLIC reconstruction which has only one polygon per cell.

3.5 Discussion

In this chapter we have presented a discrete approach to MIR based upon optimizing the labeling of fractional volume elements over a subdivided spatial domain. We introduced a *volume conservative swap* move for the optimization process, and discussed methods for extracting and visualizing material interfaces from a labeled discretization. Our technique gives significantly better results than previous methods, producing interfaces between multiple materials that are continuous across cell boundaries for time-varying and static data in arbitrary dimension with bounded error.

There remains, however, future work to be performed on the algorithmics of discrete multi-material interface reconstruction:

Thin Interfaces Our method, like most others, can have difficulty reconstructing thin interfaces. Figure 3.7 is an example. In our approach, reasonable discretization resolutions can be insufficient to allow thin surfaces to connect across a cell. Additionally, optimization of the

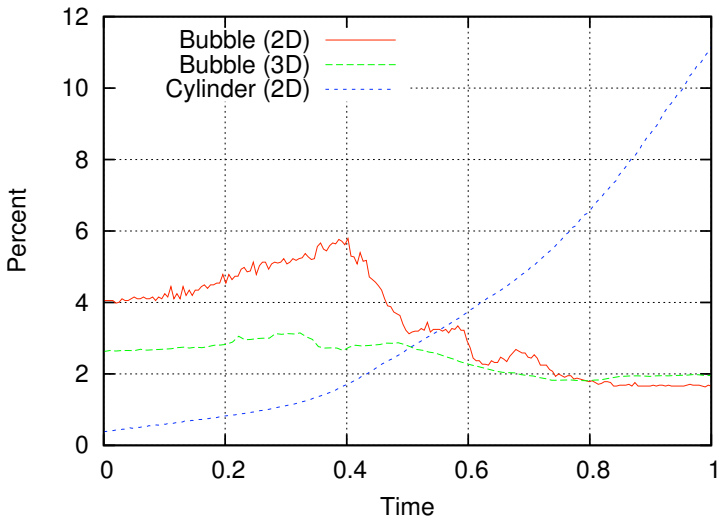


Figure 3.6: Percentage of mixed cells over time. Low percentages yield large memory savings, since subdivision is only performed upon mixed cells.

Potts-model energy can produce "blobs" rather than thin interfaces due to its surface area and curvature minimizing properties.

Energy Metric The Potts-model energy is not the only possible metric for the energy of a labeled discretization. In the presence of domain-specific knowledge, such as specific material properties, other energy metrics might produce better results than the Potts-model energy.

Smoothing As mentioned in Section 3.2, smoothing the labeling field prior to segmentation avoids unwanted visual artifacts at the sub-cell level caused by discretization. After smoothing, however, the error bound discussed in Section 3.1 is no longer valid. Empirically we have found that the small size of subcells limits the volume changing effects of smoothing: for instance, the average volume error for the three-dimensional bubble shown in Figure 3.5 was approximately 7.5% after Gaussian smoothing of the labeling field. Increasing the discretization resolution to 8^3 subcells per cell approximately halves this error.

While the volume changing effects of smoothing are relatively small and difficult to perceive during visualization, it would be best to have the advantages of smoothing while maintaining a tight error bound. In the next chapter we introduce a new material interface algorithm that incorporates new topology generation techniques with volume-adaptive surface smoothing.

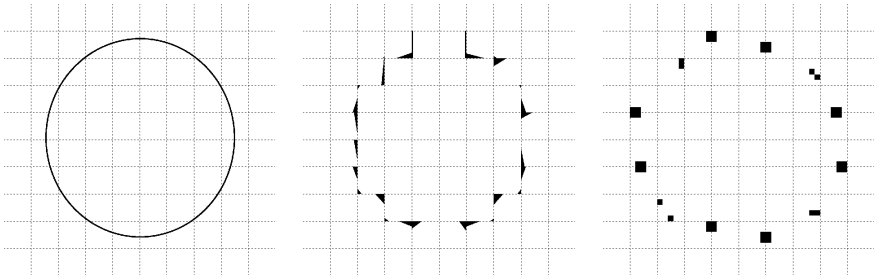


Figure 3.7: Thin interfaces (left, 10x10 grid) are difficult to reconstruct for both PLIC (middle) and our method (right).

Chapter 4

Smooth, Volume-Accurate Material Interface Reconstruction

In this chapter, we introduce a new material interface reconstruction method based upon a volume-adaptive active interface model that adjusts an initial boundary approximation toward a better solution. Our algorithm consists of two general stages:

- We generate the topology of the material interfaces using a pipeline that includes a rule-based examination of mixed cells, and a robust discretization method for ambiguous regions; and,
- We employ a *volume-adaptive* active interface model that balances internal curve/surface metrics with external volume objectives to iterate the interfaces toward a low-error solution.

Section 4.2 begins by describing some of the difficulties of topology initialization: each mixed cell must be segmented into a number of homogeneous material regions based upon its volume fractions; adjacent pure cells of different materials must be separated; and interfaces should be continuous across cell boundaries. Further, the under-determined nature of this problem that leads to topological ambiguity must be addressed. We detail a robust topology generation pipeline which satisfies these requirements for two- and three-dimensional problems with multiple materials.

Unlike previous methods that force an interface topology and sacrifice accuracy (e.g., by recasting MIR as an isosurfacing-like problem), our approach largely decouples topology generation from volume preservation. The initial interface topology that we generate is simply a starting point from

which we can improve our MIR solution.

From the initial cell-level interface topology, we create a piecewise linear surface mesh, as described in Section 4.3. We then develop a *volume-adaptive* active interface model in Section 4.4 to update the location of the mesh control points such that the interfaces approach a low-error solution to the MIR problem. At the heart of this model are smoothing and volumetric forces that iteratively adjust the surface mesh, simultaneously attempting to satisfy the volume fractions prescribed by the initial problem while improving the mesh quality. Results for 2D and 3D problems are presented in Section 4.5.

4.1 Dynamic Surface Models

Active contours – or *snakes* – were first proposed by Kass et al. [42]. These contours are parametric curves constructed with an energy functional that achieves a minimum value near a "boundary." Snakes have been utilized in numerous applications due to their segmentation capabilities [53], and in three dimensions snakes become an alternative surface reconstruction method. Cohen and Cohen [16, 17] published the initial extensions to achieve active surfaces – called *balloons* – and used them in three dimensions to extract facial skin from MRI volume data. Takanashi et al. [77, 78] directly extended the definition of snakes from curves to surfaces, creating an "active net" method, which was used to segment and extract muscle tissue from the visible human dataset. Gibson [28] uses a three-dimensional surface net to segment binary data smoothly. Ahlberg [3] also provides a good discussion of the extension of active contours to three dimensions. In this chapter, we use the term *active interface* to mean an explicit, dynamic surface in either two or three dimensions.

The level set method, introduced by Osher and Sethian [61], represents dynamic interfaces as the zero level set of a time-dependent, implicit function. By solving the equations of motion in an adaptive, narrow-band Eulerian grid [1], the level set method is a computationally efficient solution for a number of fluid simulation, surface reconstruction, and segmentation applications – especially where surface topology change is required (see [69, 70, 62, 60]). Like active contours, the level set method scales well from two to three dimensions, and recent work has focused upon handling

multiple spatial regions or fluids; in particular: by using one level set per region (see [55, 66, 72, 49]), or combinatorically using n level sets to represent 2^n material regions as done by Vese and Chan [81].

Both explicit and implicit dynamic surface models have been widely used in a range of applications. We have elected to use mesh-based active interfaces [3]. As we show in Section 4.4, an explicit scheme allows for straightforward calculations of local forces, and updates to the current per-cell reconstruction error; moreover, meshed surfaces are ideally suited for later use in the visualization and analysis pipeline.

4.2 Material Interface Topology

Generating the initial surface topology for a volume fraction problem comprises the first stage of our reconstruction process. The *topology* of a surface describes its fundamental shape, such as the number of components and holes in the surface. The *embedding* of the surface refers to a geometric representation with specific spatial location, often represented explicitly as a mesh of vertices and faces. In the MIR problem, our initial focus is on generating cell-level interface topology – i.e., the general configuration of boundary surfaces within each cell – for mixed cells in two and three dimensions.

Consider a cell with two materials A and B: possible, valid topologies might range from a simple continuous interface between A and B, to multiple disconnected "islands" of A within B. The embedding of the continuous interface topology could be linear or curved, and the islands might be round or ellipsoidal. These are unknowns that make material interface reconstruction difficult: for any mixed material cell, there are limitless topologies and embeddings that might satisfy the volume fractions.

Figure 2.1 of Chapter 2 shows four possible reconstructions for the same volume fraction data. Our solution to the topology problem is a pipeline of three topology generation methods to initialize per-cell material interface topology. In the first pass, we use the *boundary* method to extract coarse,

mesh-aligned interface topology. Next, the *rule-based marching* method – which can handle common binary- and multi-labeled segmentation cases – is applied across the mixed cells of the volume fraction data. This stage initializes the topology in cells containing few materials and simple interface configurations. Finally, the *discretized boundary* method is applied in complex regions where ambiguity caused by fine-scale features or many materials makes case-table analysis difficult. Figure 4.1 provides a working example of this pipeline.

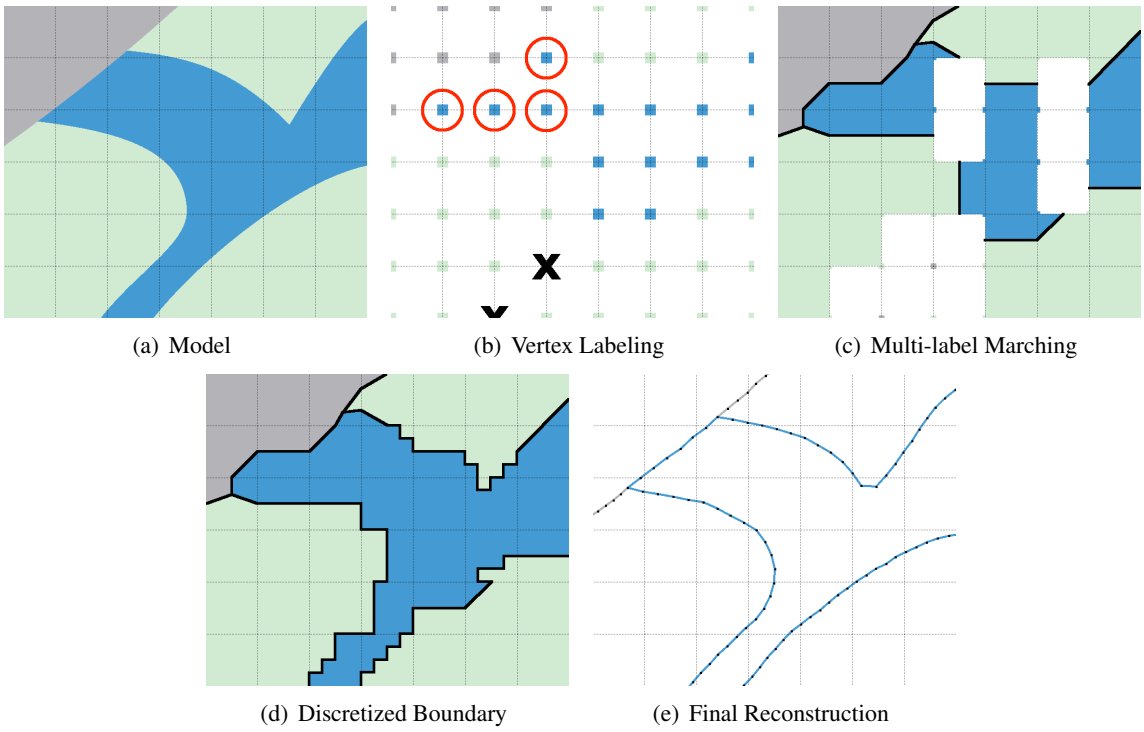
4.2.1 Boundary Method

Our topology generation pipeline starts with the extraction of the most basic material interfaces: those between pure cells of different material. Given two neighboring cells c_1 and c_2, with volume fractions of 1.0 for materials A and B, respectively, the shared face f between the cells is part of the A-B material interface. Such interfaces often arise in simulation, where it is common to initialize each cell as a homogeneous material; mixed cells are then the result of advected material interfaces [33, 64]. These trivial boundaries are often required to form complete, closed material regions, and extracting them at this stage simplifies the next stage of the topology generation pipeline.

4.2.2 Rule-based Marching Method

While MIR is a fundamentally different problem than isosurfacing, it is often the case that simple topologies are sufficient for interface reconstruction. In many simulation codes, for example, a goal is to have approximately linear interfaces within each cell [36]; correspondingly, the average cell size is set (or adapted) based upon the expected interface curvature and complexity. In this situation, the vast majority of cells in a volume fraction dataset will either have no interface, or relatively simple interfaces between a few materials. In this section, we develop a rule-based "marching"-style segmentation method to generate cell-level interface topology based upon this observation.

The *rule-based marching* method applies a small set of rules to a problem's volume fractions in order to derive a partial vertex labeling, along with edge and cell characteristics. From this labeling,



(a) Model (b) Vertex Labeling (c) Multi-label Marching

(d) Discretized Boundary (e) Final Reconstruction

Figure 4.1: The first step toward a material interface solution is the generation of initial surface mesh topology. In (a), we show an example 3-material problem. The *rule-based marching* method is used to (b) label the vertices and (c) fill in topology from binary- and multi-label segmentation lookup tables. In (b) the majority of vertices were labeled using the **pure cells** vertex labeling rule, however the circled vertices were labeled with the **neighborhood** rule, and the two vertices marked with an "X" could not be labeled using our rules. In (c), topology has been filled from the case table shown in Figure 4.2; blank cells could not be initialized due to ambiguity. The *discretized boundary* method is applied in (d) to generate topology for the remaining, uninitialized cells. In (e) we show our proposed reconstruction after applying volume-adaptive active interface optimization.

it is possible to derive material interface configurations from binary- and multi-label segmentation lookup tables. One of the primary differences between our *rule-based marching* method and previous isosurfacing approaches for MIR is that vertex labels are implied, rather than forced, from the volume fraction data. Bonnell et al. [10, 9] cast volume fraction data onto a dual mesh, while Meredith [54] forces vertex labels through volume averaging. Our approach, on the other hand, is to construct a partial labeling through the application of labeling rules; cell-level topologies from a lookup table are only used when they are suitable in the context of the problem's volume fractions.

Consider the labeling L, where $L(v)$ is the label of vertex v. In a complete labeling, each vertex in an n material problem has a known material label in the set $\{1,\ldots,n\}$. Partial labelings, however, allow vertices to have an unknown material:

$$L(v) = \begin{cases} 1,\ldots,n & \text{if the material at vertex is known, or} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

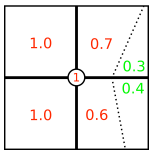
Volume fraction data is without vertex information, and thus the initial labeling of each vertex is undefined.

The first step of the *rule-based marching* method is to label mesh vertices using the volume fraction data. For each vertex in the mesh we evaluate the following rules:

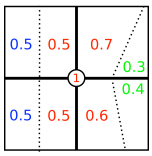
Pure Cells If all pure cells neighboring vertex v are of material i, then $L(v) = i$; if there are neighboring pure cells of different material, however, then $L(v)$ is undefined.

Neighborhood Consider the set of materials M common to all (pure and mixed) cells neighboring vertex v. If M contains a single material then the vertex label is set to that material.

These vertex labeling rules are illustrated for the central vertex in the following diagrams (where material one is red, and the dashed lines are hypothetical interfaces):



Pure Cells



Neighborhood

The **pure cells** rule is designed to "push" the material of pure cells onto neighboring vertices, without promoting arbitrary material precedence. (Note that interfaces between pure cells have already been extracted by the *boundary* method.) The **neighborhood** vertex labeling rule is useful for labeling vertices in the presence of thin filaments and shells, where the feature in question spans one or more vertices but not an entire cell.

From vertices, we move on to the edges of the mesh. Consider a mesh edge e between vertices v_1 and v_2. In typical marching schemes, the number of interface crossings χ along e is known *a priori* based upon the vertex labeling: edges with matching labels have no intersection ($\chi = 0$), while edges with mismatched labels have one intersection ($\chi = 1$). In the MIR problem more flexibility is required; thin filaments, shells, fine-scale features, and multiple materials can produce *multiple* intersections along a single mesh edge. Here, edges with matching vertex labels are either not intersected, or intersected *more* than once ($\chi = 0$ or $\chi \geq 2$); edges with mismatched vertex labels must have one or more intersections ($\chi \geq 1$).

Edge labeling rules are used to determine – to the extent possible – the number of interface crossings along each mesh edge. In our work, we use the following pair of rules:

Pure Cells The edge e has no intersections if any neighboring cell along e is pure.

Neighborhood Consider the set of materials M common to all (pure and mixed) neighboring cells along the edge e. If M contains a single material then e has no intersection.

Largely analogous to the vertex labeling rules above, edge labeling rules are based on the observation that edge intersections are building-blocks for defining material regions that span cells; i.e., if an edge is intersected by an interface, then there are at least two materials along that edge and those materials must be in present in the neighboring cells. The **neighborhood** rule can also be used to

derive certain vertex labels: if edge e (connecting vertex v_1 to vertex v_2) has no intersections, then the labels of its end vertices must match; i.e., $L(v_1) = L(v_2)$. For three-dimensional problems these rules translate naturally to cell faces as well.

Once we have applied the above rules we are left with a partially labeled mesh. Figure 4.1(b) shows the results of this labeling process when applied to the volume fractions derived from the three-material model problem shown in 4.1(a). The majority of vertices were labeled using the **pure cells** vertex labeling rule, however the circled vertices were labeled with the **neighborhood** rule, and the two vertices marked with an "X" could not be labeled using our rules.

The next step in the *rule-based marching* is to generate per-cell interface topology. As with other marching-style segmentation methods, we iterate over each cell in the mesh incrementally adding surface fragments to our material interfaces. To find the appropriate surface fragment(s) to add to the interface for each cell, we perform a lookup using the cell's labeling configuration in a binary- or multi-label segmentation table.

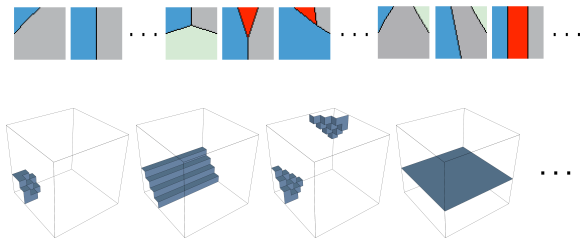


Figure 4.2: Partial 2D (top) and 3D (bottom) case tables for per-cell interface generation used in the *rule-based marching* method.

Consider the cell c. If c has an unlabeled vertex v – i.e., $L(v)$ is undefined – then the cell topology is left uninitialized. Otherwise, if all vertices are labeled, then we perform a lookup in the segmentation case table. In most cases, the returned surface fragment(s) from the case table will properly segment the cell into enough regions to provide for a valid MIR solution. Unlike typical binary- and multi-label segmentation, however, two situations might occur: the case table might indicate ambiguity, or more materials might be needed. In both of these situations the cell's interface topology is left uninitialized. To test if more materials are required, let M be the set of materials with nonzero volume fractions in cell c, and let M' be the set of material labels for the vertices of the cell. If $M \neq M'$ then c has "residual" material. The implication is that there are either internal "pockets" of

material within the cell or there are multiple intersections along one or more of the edges (or faces) of the cell.

Figure 4.2 illustrates the case tables we use to generate interfaces in 2D (top) and 3D (bottom). We have created our 2D table to allow multiple intersections along mesh edges, and for extra material regions within cells in certain configurations – e.g., a triple point where one edge of the cell is intersected twice. In the 2D case table in Figure 4.2, residual material regions have been colored red. We also mark cases in the table as "ambiguous" if there are multiple valid ways to segment a labeling (consider the ambiguity addressed by Nielson and Hamann for marching cubes [58], and the complexity introduced by allowing multiple intersections per edge). To construct the 2D table we use a recursive scheme to label vertices, intersect edges, and connect intersections. Our 3D case table is discussed further in Section 4.3.

To generate cell-level interface topology in this stage, the cell's vertices must be labeled, the case table must not indicate ambiguity, and the materials provided by the returned segmentation must match the cell's material requirements. Figure 4.1(c) shows the result of *rule-based marching* over our running 2D example. Interface configurations typically seen in isosurfacing problems, as well as some "T" junctions between three materials have been captured. Several cells remain uninitialized, however, either as a result of uncertain vertex labels, or because an ambiguous choice would have been required given the available segmentation topologies. In Figure 4.1(c), blank cells represent those that could not be initialized; the final stage of our pipeline, however, is able to generate topology in these cells.

4.2.3 Discretized Boundary Method

We solve for the topology of any remaining uninitialized cells using a *discretized boundary* method. In this stage of the topology generation pipeline, we apply the discretized labeling method described in the previous chapter on a per-cell level. Figure 3.1 from Chapter 3 provides an overview of this topology generation method. Cells that have been left uninitialized by previous stages in the pipeline are subdivided into small, fractional volume elements. These "subcells" are then labeled

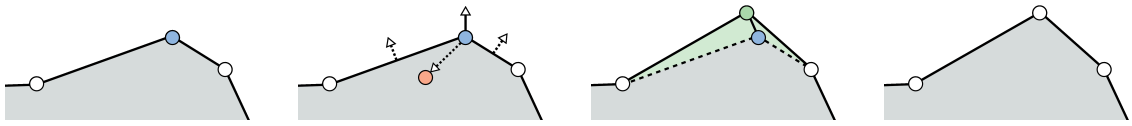
by a material, where the number of subcells with label i is proportional to the volume of material i within the cell. Next, we perform energy minimizing optimization to improve the labeling; by using a "quenched" Potts-model energy, the labeling energy will monotonically decrease and rapidly converge. Here, "quenched" refers to simulated annealing with a 0 temperature (see Chapter 3 for details). Once the labeling has converged, coincident faces between subcells with different labels are extracted to become part of the initial material interfaces. Figure 4.1(d) illustrates the use of the *discretized boundary* method to generate topology within ambiguous cells left over from the *rule-based marching* method.

4.3 Initialization

At this point in the reconstruction process, each mixed cell has been attributed an initial cell-level interface topology. Before applying our volume-adaptive active interface model to improve the topological embedding, however, a mesh-based representation of the interfaces must be created. To represent the interface, we use a mesh data structure with vertex-face incidence information (see [26]), which trivially allows cell-level interface topologies to be merged into a single interface.

We begin by adding the faces produced by the *boundary* and *rule-based marching* methods to our interface mesh; the faces from these methods are continuous across cell boundaries by definition. Next, we construct a second closed, mesh-based representation of sub-cellular faces produced by the *discretized boundary* method. Finally, we take the union of the two meshes, and discard faces that would create a partition between regions of the same material. Figure 4.1(d) shows material interfaces (in black) that result from combining interfaces from the *rule-based marching* and *discretized boundary* methods.

In two-dimensions it is straightforward to join the interfaces generated by the *rule-based marching* and *discretized boundary* methods. In 3D, however, merging the meshes can be more difficult; to obtain a crack-free interface significant re-meshing might be required. To simplify our implementation, our 3D case table is constructed to return surfaces that can be matched easily to the meshes produced by the *discrete boundary* method. This construction is based upon the description by Hege



(a) Interface (b) Forces (c) Swept Volumes (d) Subsequent Interface

Figure 4.3: An interface point is updated within our volume-adaptive active interface model. Consider the blue control point in (a), part of the interface between the empty white region and the shaded grey region. We use local forces – shown in (b) – to update the position of the control point. The *smoothing force* is the offset needed to move the control point to the average of its neighboring control points (shown in orange); the *volume force* is the average of the oriented normals for the faces surrounding the current control point. After moving the control point in (c), we update the volume of each material region by analyzing the area swept by the control point (shaded green). As this update step is repeated for other points on the interface the reconstruction improves.

et al. [31]. While the material interfaces will be "blocky" at this point, they will be optimized in Section 4.4.

Two bookkeeping tasks end the initialization:

- Every surface mesh face – i.e., segment in 2D, or triangle in 3D – is marked with the two materials that it segments. This makes it simple to determine the orientation of mesh faces and the material regions that they bound.
- We next calculate the per-cell, per-material volume prescribed by the initial surface mesh. The *boundary* method separates only pure cells, and thus no volume calculation is necessary; in the *rule-based marching* method we precompute the per-material volume of each cell segmentation during the lookup table initialization; and in the *discretized boundary* method, summation over the discretized labeling yields the correct volume for each material.

4.4 Volume-Adaptive Active Interfaces

In this section, we develop a *volume-adaptive* active interface model that iteratively refines the initial interfaces created in Section 4.2 toward a better reconstruction. The basis of our model is to deform the material interfaces under the influence of local forces. At each iteration step we pick a random control point, and move that point to satisfy our model's objectives better: surface quality

and volume accuracy. We continuously update the volume error of the material interfaces, and discuss methods to monitor and enhance convergence.

Active interfaces are represented as piecewise linear curves or surfaces composed of a set of k control points $P = \{p_1, \ldots, p_k\}$ connected by line segments or triangles. Our volume-adaptive model defines two local forces that can be computed per-vertex in the interface mesh. The first is an internal *smoothing force* that attempts to reduce the curvature of the mesh and make control points equidistant; the second is an external *volumetric force* normal to the surface that adjusts the mesh to better fit the volume fraction data. Both forces are computed directly from local interface mesh information, and thus scale with the mesh.

Laplacian smoothing [25] is widely used to approximate internal surface forces within active interface models [3, 28]. The smoothing operation is performed by iteratively replacing a point by the average of its neighboring vertices. Given a sequence of points $p_1, \ldots, p_n$ representing a piecewise linear curve, Laplacian smoothing at point i replaces p_i by the average of p_{i-1} and p_{i+1}. In the case of a surface in three dimensions, p_i is replaced by the average of the points in the 1-ring of point i.

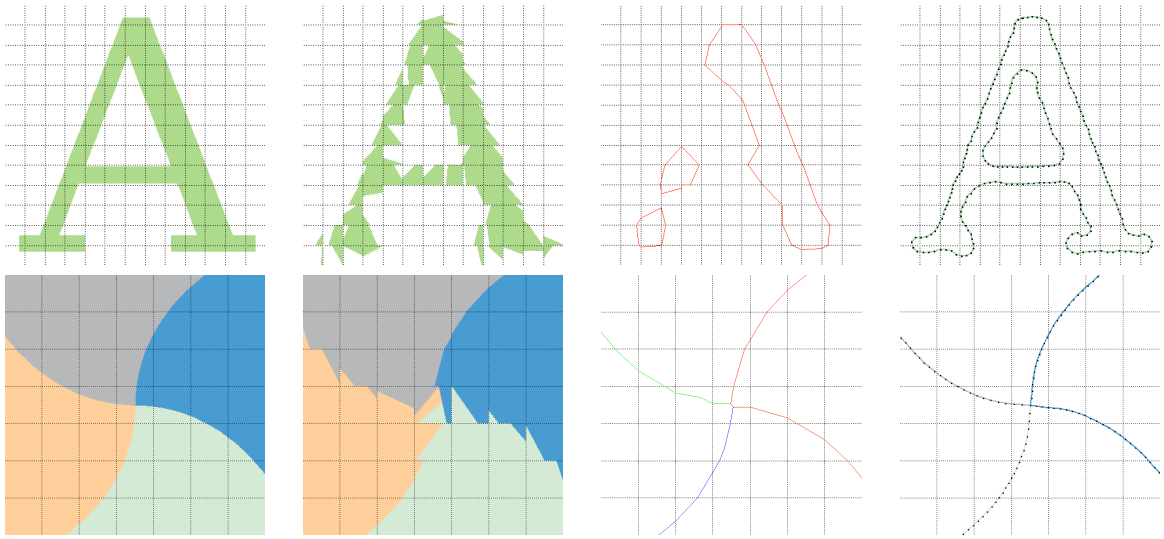
We use Laplacian smoothing to determine the internal force pushing on each point of the interface. Let p_i be a control point defining the piecewise linear interface, then the force is determined to be

$$F_{\text{int}} = d - p_i,$$

where d is the average position of the control points that neighbor p_i.

One difficulty with Laplacian smoothing is the fact that it tends to shrink a curve. Historically, shrinking has been counter-balanced by an external force normal to the curve (see Cohen and Cohen [16, 17]). In a material interface context, however, the shrinking of a curve bounding one material region corresponds to an increase in other materials' volumes. This observation motivates the addition of an external force that does not simply try to increase every bounded region's volume, but instead adaptively pushes the interface toward a better local match to the volume fractions.

Let $\varepsilon(c, A)$ be the signed error between the known and reconstructed fractional volume of material



(a) Model (b) PLIC (c) Meredith (d) Proposed

Figure 4.4: Two 2-dimensional volume fraction problems: the letter "A", and a four-material junction between intersecting curved interfaces. We show the model interfaces in (a) from which we calculate volume fractions for reconstruction. The PLIC reconstruction of these problems is shown in (b), an isosurfacing-like reconstruction using the method by Meredith is shown in (c), and our proposed reconstruction method in (d).

A within the cell c containing control point p_i. If the signed error is positive, then there is too much volume assigned to material A, and surfaces bounding material A within the cell should shrink to reduce the volume. On the other hand, when the error is negative, the bounding surface should grow such that A is awarded more volume within the cell.

To capture this desired behavior, we define the volume-adaptive external force at p_i for material A to be the oriented average of the normals of the faces surrounding p_i:

$$F_{\text{ext}}(A) = \left(\varepsilon(c,A) \sum_{f \in R_A} \perp_A (f) \right),$$

where R_A is the set of faces surrounding p_i that are marked with material A in cell c, and $\perp_A (f)$ is the normal of face f oriented into material region A. Multiplying by $\varepsilon(c,A)$ adaptively orients the external force to shrink or grow the region bounding material A as needed to better match the problem's volume fractions.

We now combine these forces – illustrated in Figure 4.3(b) – to update the positions of points that

define the piecewise linear material interfaces. Consider a control point p_i within the mixed cell c. At p_i there is a single smoothing force, but there will be multiple volumetric forces: two if the control point's neighborhood is a topological disk separating two materials; more if the neighborhood is multi-material junction. We address this issue stochastically in our active contour model by only considering a single material's volumetric force per iteration.

For a randomly chosen material ξ, the total local force at p_i becomes:

$$F = \alpha F_{\text{int}} + \beta F_{\text{ext}}(\xi),$$

where α is the weighting of the internal force within the active interface model, and β is the weighting of the external force. Finally, we update the position of the control point:

$$p_i' = p_i + F.$$

After moving a control point it is necessary to update the current error of the reconstruction ε in order to maintain the accuracy of the volume-adaptive force. We calculate the volume swept by the 1-ring of p_i as it moves to p_i': in two dimensions line segments are swept to form triangles as shown in Figure 4.3(c); triangles are swept into tetrahedra in three dimensions. The swept triangles or tetrahedra are then clipped against mesh cells in the local neighborhood, and the oriented volume of the clipped triangles or tetrahedra is used to update the current error of the reconstruction. Incrementally maintaining material volumes is much faster than recomputing the entire volume each update.

To obtain useful results with any active interface model, the forces within the system must be balanced. In our implementation, a simple yet effective rule is to set the weight of the internal force proportional to that of the external force

$$\alpha \propto \beta,$$

and then monotonically decrease α and β at a rate relative to the change in error until the material interfaces converge. Balancing the weights in this manner is analogous to gradually reducing the

temperature parameter of simulated annealing [43].

Finally, we note that in order to obtain both volume-accuracy *and* smoothness it can help to subdivide large faces on the boundary surface. A threshold parameter σ is introduced to control the maximum surface face size – i.e., maximum line segment length or triangle area. Using a very large value for σ will ensure that subdivision is not used in cases where coarser surfaces are desired.

4.5 Results

This section evaluates our method – and compares it to existing methods – over multiple volume fraction datasets. We have generated synthetic data in two and three dimensions to compare against model reconstruction solutions. In three dimensions we present two real-world volume fraction examples. The first example is from a CFD simulation using the Volume-of-Fluid (VOF) method. The second is from an Embedded Boundary (EB) problem. In EB problems, a fixed boundary surface is represented using volume fractions – rather than geometry – in order to facilitate multi-physics simulation around the boundary. Results were obtained on an Apple MacBook Pro notebook computer with a 2.33 GHz Intel Core 2 Duo processor and 2 GB of memory.

As a reference, Table 4.1 provides summary information about the datasets and our reconstructions, including: number of materials, dataset size, percent of mixed cells, number of faces in the reconstructed interface, and convergence time. Table 4.2 plots on a logarithmic scale the average reconstruction error of our method, the isosurfacing-like reconstruction of Meredith [54] (as implemented in VisIt [15]), and the discrete method of Anderson et al. [5]. Error is computed as the average maximal volume fraction differential between a reconstruction and the problem's known volume fractions:

$$E = \frac{\sum_c \max_{i \in (1,\ldots,n)} |\varepsilon(c,i)|}{|\text{Mixed Cells}|}.$$

For example, an error of 0.01 indicates that the expected maximum misclassification of material in each mixed cell is 1% of the cell's volume.

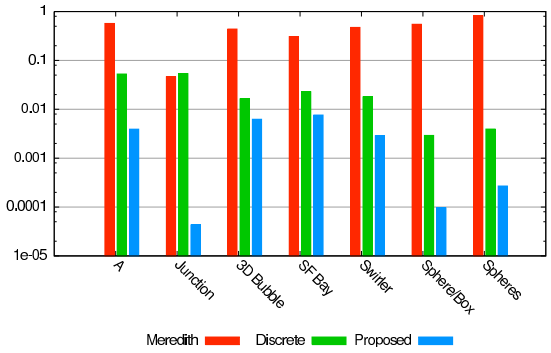
Table 4.1: Problem sizes, surface complexity, and reconstruction times.

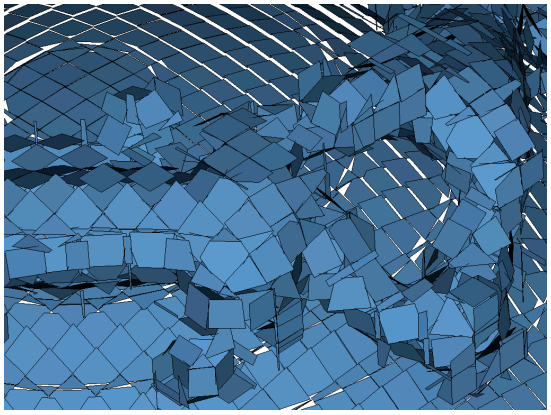
	Materials	Extents	Mixed	Faces	Time (m:ss)
A	2	20^2	17.5%	242	0:10
Junction	4	7^2	42.86%	130	0:13
3D Bubble	2	64^3	2.65%	143,640	3:42
SF Bay	2	$258^2 \times 10$	1.99%	117,610	5:17
Swirler	2	64^3	6.13%	356,904	4:24
Sphere/Box	3	13^3	15.66%	15,014	3:02
Spheres	6	13^3	58.03%	73,392	2:56

Our first set of tests are over two-dimensional synthetic volume fraction data for which we have a model solution. We compute the volume fractions from the model images in 4.4(a), and attempt to reconstruct the boundaries using PLIC [86], the method of Meredith [54], and our proposed reconstruction method. The top row in Figure 4.4 shows reconstructions of a letter "A" in serif font embedded within a 20^2 cell grid. Our topology generation pipeline correctly captures the thin filament structures, and our volume-adaptive active interface model iterates the boundary to an average mixed cell error of only 0.3924%. The bottom row of Figure 4.4 shows reconstructions of a problem in which multiple curved interfaces intersect at a 4-material "junction." This problem also requires the generation of non-trivial topology. PLIC does a poor job of reconstructing this interface because the correct topology cannot be represented by manifold, binary segmenting surfaces. The *rule-based marching* method is able to extract the curved 2-material interfaces, although the center cell containing four materials remains uninitialized under that method. Running the *discretized boundary* method upon the center cell produced the correct topology. Another valid topology that is occasionally generated by the *discretized boundary* method has two 3-material junctions, similar to the reconstruction produced by the method of Meredith [54]. Our final reconstruction has an average mixed cell error of 0.0043%.

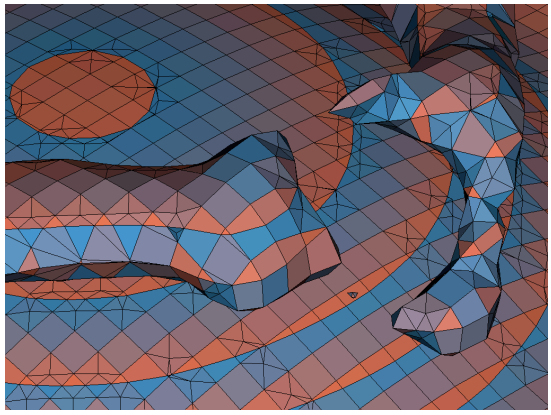
Next, we consider a three-dimensional Volume-of-Fluid simulation of a low density bubble rising through a denser fluid. The computational domain was 64^3. After the bubble reaches the surface, it bursts as a result of surface tension. Figure 4.5 provides a closeup view of this dataset after the bubble has burst: in 4.5(a) we show a 3D version of the PLIC algorithm, the result of which is a set of disconnected polygons that partition each mixed cell into two material regions; in 4.5(b) we show the reconstruction by Meredith [54]; and in 4.5(c) we show the surface mesh generated by

Table 4.2: Log plot of average per-cell error for various methods.

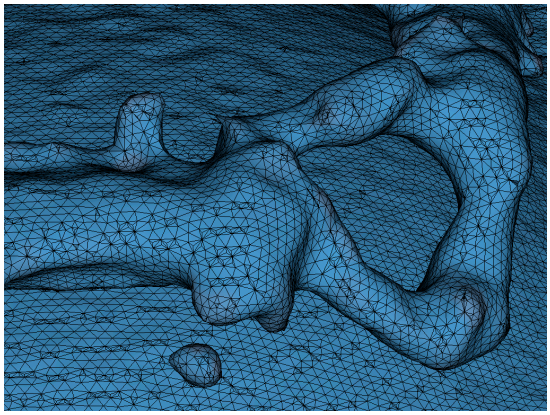




(a) PLIC

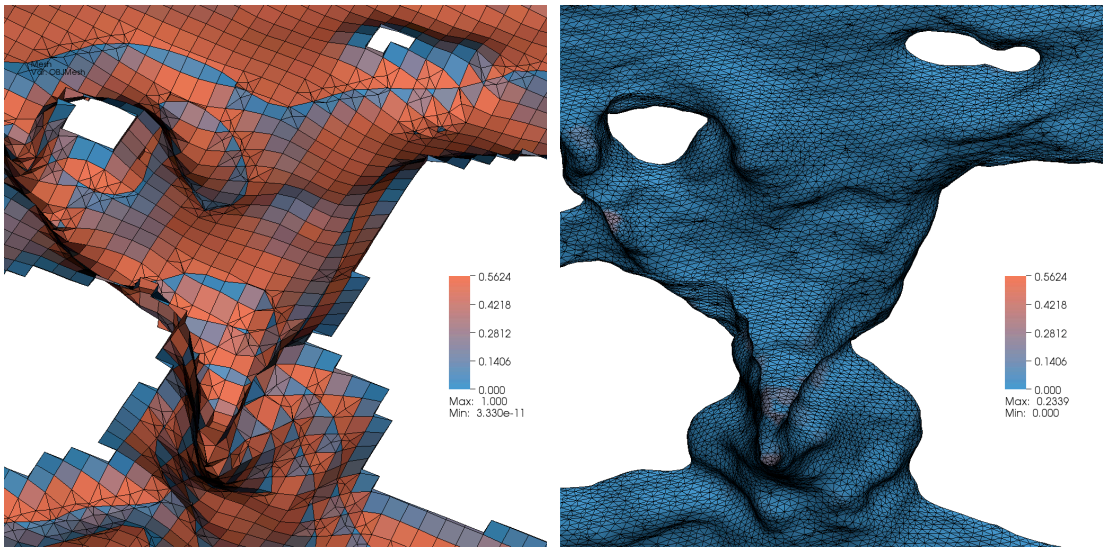


(b) Meredith



(c) Proposed

Figure 4.5: Reconstructions in three-dimensions of a low-density bubble of fluid rising through a higher density fluid after the bubble has burst through the surface. In (a), we show the PLIC interface reconstruction; (b) shows an isosurfacing-like reconstruction using Meredith's method; (c) is our proposed reconstruction. Interfaces have been pseudocolored by the per-cell reconstruction error.



(a) Meredith (b) Proposed

Figure 4.6: Reconstructions of an embedded boundary (EB) representation of bathymetry – or underwater depth – data from the San Francisco Bay using: (a) Meredith's method, and (b) our active interface reconstruction method. Interfaces have been pseudocolored by the per-cell reconstruction error.

our proposed reconstruction. Pseudocolor has been used to visualize the local, cell-level error on reconstructed interfaces in 3D problems. A blue-to-red colormap is used for the range of $[0.0, 1.0]$ volume error: blue indicates low error with little misclassification of material, while red indicates higher error.

We now turn to a three-dimensional embedded boundary representation of bathymetry – or underwater depth – for the San Francisco Bay. This type of data can be used in a wide range of simulations, from ocean currents and tidal flow to modeling oil spills. The dataset is a $258^2 \times 10$ rectilinear grid with one explicit volume fraction, representing two materials: above and below the boundary. Figure 4.6(a) shows the reconstruction produced by the Meredith's algorithm [54], while our proposed reconstruction is shown in 4.6(b). The colormap in this figure is also blue-to-red and indicative of volume error, but the range is $[0.0, 0.5624]$. Our reconstruction produces smoother interfaces with much lower average per-cell error: 0.7640% per mixed cell for our method, versus 43.8410% for the method by Meredith [54].

Next, we consider reconstructing a "low-swirl burner." Low-swirl combustion is an aerodynamic

flame stabilization method that produces ultra-lean flames with low emission, and is largely used for industrial heating and gas turbines [14]. The embedded boundary dataset we use is a 128^3 rectilinear grid of volume fractions derived from the constructive solid geometry definition of a low-swirl burner. Our reconstructions focus upon the lower 64^3 octant of the full dataset due to the symmetric nature of the geometry. Figure 4.7 compares the reconstruction by Meredith's method (left) to our proposed reconstruction (right); notice that our method fully captures the swirler blades and central screen. In our reconstruction, the *rule-based marching* method generates interface topology for 71% of the mixed cells (mostly on the large cylindrical portions of the burner), while the *discrete boundary* method generated interfaces for the swirler blades and central screen.

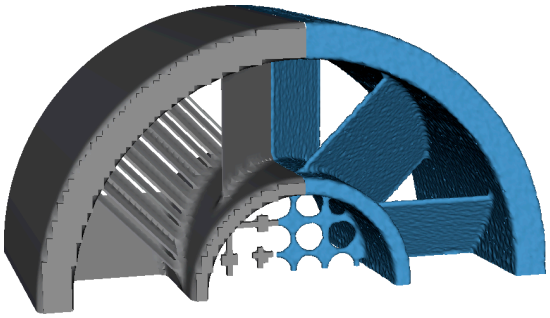


Figure 4.7: Comparison of Meredith's reconstruction (left) to our proposed reconstruction (right) for one quadrant of the swirler dataset.

Finally, constructive solid geometry has been used to create analytic test datasets in three-dimensions. Two datasets are shown in the left column of Figure 4.8: on top, a box intersected by a sphere (box: $(0.2, 0.2, 0.2)$ to $(0.6, 0.6, 0.6)$; sphere: center $(0.6, 0.6, 0.6)$, radius 0.2); on bottom, five concentric spheres (centers $(0.5, 0.5, 0.5)$, radii $\frac{1}{13}$, $\frac{2.25}{13}$, $\frac{3.5}{13}$, $\frac{4.75}{13}$, and $\frac{6}{13}$). The sphere/box problem has 3 materials, while the multiple spheres problem has 6 materials – in both problems, "empty" space is another material. In the middle column of Figure 4.8 we show the reconstructions performed using the algorithm of Meredith [54], while our proposed reconstructions are shown in the right column. Error results are listed in Table 4.2, however the figure also includes horizontal lines to help illustrate differences between the reconstructions and the problem models.

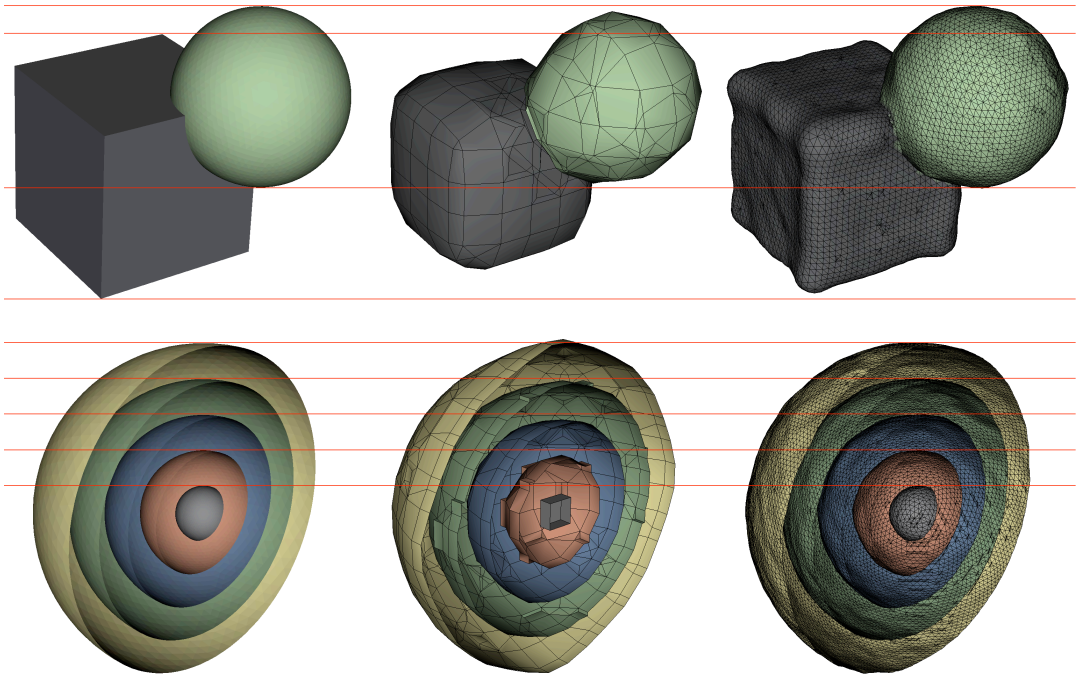


Figure 4.8: Analytic three-dimensional problems with multiple materials: on top, a box intersected by a sphere (3 materials); on bottom, five concentric spheres (6 materials). Horizontal lines help to illustrate the differences between reconstructions; note the volume loss with an isosurfacing-like approach compared to our proposed method.

Part II

Function Fields

Chapter 5

Background

With increasing computing power and the ability to gather more and more data via increasingly powerful imaging and sensor technology, we can generate data sets of ever increasing complexity. Datasets that represent physical phenomena now contain billions of multi-valued, multi-dimensional, time-varying elements, and are difficult (or impossible) to analyze by the classical scalar- and vector-field algorithms commonly used in the visualization community [48, 12, 46].

In the second part of this dissertation, we address the visualization and analysis of *function fields*, a class of high-dimensional, multi-variate data. Function fields directly arise in applications where an entire spectrum of values is simulated/collected at each data point. From hyperspectral imagery to ground cover distributions, ocean, weather, and air quality simulations, we find data in which samples do not correspond to collections of disjoint scalar values, but rather one-dimensional scalar functions:

$$F : p \in \mathbb{R}^n \rightarrow f_p \in \mathscr{F}_I,$$

where $\mathscr{F}_I$ is the set of functions over a closed interval I. Consider hyperspectral imagery, the structure of which is depicted in Figure 5.1(a); here sophisticated sensors produce images in which individual pixels correspond to sampled functions of the spectral intensity of visible and infrared light. Functions are typically represented by a discrete set of m samples over the functional domain. Furthermore, m is often large, leading to tens or hundreds of samples per data point.

5.1 Related Work

In addition to domain-specific techniques, dimension reduction, clustering, and query-driven approaches have been used for the visualization and analysis of function field data. Dimension reduction methods project high-dimensional data to fewer dimensions so that traditional visualization techniques can be applied; clustering assigns labels to data based upon some criteria; and queries explicitly segment the data by evaluating constraints upon the original, high-dimensional space.

A common approach for visualizing function fields involves casting them as scalar fields, either directly, by treating the interval I over which a field's functions are defined as an extra space or time dimension [23, 32], or through local operations. Kao et al. [41, 40] and Luo et al. [51] use parametric statistics and shape descriptors to describe functions using scalar values. For example, a two-dimensional hyperspectral image might be viewed as a three-dimensional image cube, or as a two-dimensional scalar field of averaged radiance.

Principal Component Analysis (PCA) [37] is an ubiquitous dimension reduction technique. For a set of vectors in m-dimensional space, PCA identifies a set of ordered, orthonormal basis vectors. Transforming the data vectors into a space spanned by the first $k < m$ of these basis vectors yields a dimension reduction that maximally preserves variance. PCA has been used to display hyperspectral imagery by associating components with color channels to produce color images [79, 34].

Multidimensional scaling (MDS) [18] may be used to embed high-dimensional data samples in a low-dimensional metric space, such that similar samples are close and dissimilar samples are distant. Once MDS has been performed, the low-dimensional space may be visualized (e.g., by using software such as Voromap [63], or as by Fang et al. [24]) to study the similarity structure of the original data. Spatial datasets, such as function fields, are ill-suited to MDS visualization, however, since the original spatial layout of the data is lost.

The extra dimension inherent to function fields can often be eliminated through domain-specific specialization. In hyperspectral imagery, for example, each pixel may be colored by integrating the radiance versus wavelength functions with color matching functions, such as CIE XYZ, which mod-

els the wavelength-dependent response of the human eye [85], or the spectrally weighted envelopes of Jacobson and Gupta [34]. Furthermore, hyperspectral imagery can be processed using linear spectral unmixing [71] to estimate the ratios of material within each pixel. Information theoretic approaches have also been presented to optimize band selection in spectral images [4, 73].

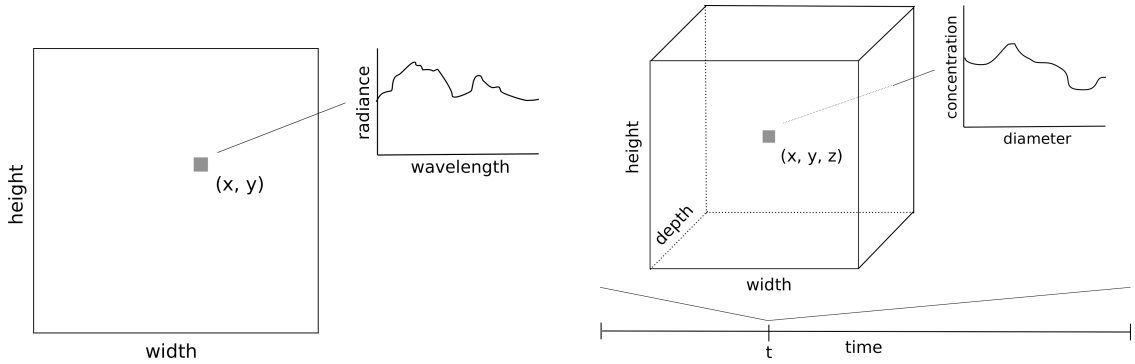
Recent work has focused on using distance or similarity measures to perform dimension reduction of function fields for visualization. Chapter 6 derives scalar fields from function fields by computing function-space distance to a "probe" within the data. Fang et al. [24] present a similar approach for visualizing time-varying data from medical imaging sensors using both function-space and geometric distance measures. In a similar vein, clustering techniques such as k-Means and Vector Quantization [2, 52] may also be applied to segment and visualize function field data, however high dimensionality can lead to poor clustering results [35].

5.2 Datasets

Function fields arise in many application domains. In Chapters 6 and 7 we visualize and extract features from two types of function fields: hyperspectral imagery, and particulate pollution data.

Hyperspectral imaging systems are used in remote sensing for a broad range of applications, including environmental studies and military preparation. The primary benefit of using a hyperspectral imagery system, such as the Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) [80], is that each pixel contains data for multiple spectral channels (instead of only grayscale or RGB), thus allowing more in-depth image analysis.

Function fields also arise from computer simulation of air quality models, such as the California Regional Particulate Air Quality Study. The CRPAQS study is concerned with particulate pollution throughout the San Joaquin Valley, California, U.S.A. During computer simulation, particulate pollution concentrations are tracked within a three-dimensional atmospheric volume over time. Because the size of aerosol particles is an important factor in their toxicity and behavior, particulate concentrations within each cell are represented as a sampled function of particle concentration ver-



(a) Hyperspectral Image (b) CRPAQS Dataset

Figure 5.1: Graphical overview of the two function field datasets used in this dissertation. In (a), we show the data layout of hyperspectral images. Hyperspectral images are spatially two-dimensional with pixels that are sampled functions of radiance (or reflectance) versus wavelength. In (b), we show the data layout for the California Regional Particulate Air Quality Study (CRPAQS) dataset. Each cell in this time-varying, three-dimensional function field contains a sampled function of particle concentration versus diameter.

sus diameter. The datasets produced from these simulations are time-varying, three-dimensional function fields as illustrated in Figure 5.1(b).

Chapter 6

Interactive Visualization by Range-Space Segmentation

We approach the visualization and analysis of function fields by creating range-space segmentations of the function field data. To begin, we define a similarity metric over the space of functions. Next, a list of *function samples* within the data domain is created – guided by application-specific knowledge, data statistics, or by directly manipulating a spatial probe. These samples are used to compute a range-space segmentation of the data. From such a segmentation, we are able to generate meaningful visualizations, and also extract separating surfaces between features.

We visualize these range-space segmentations by defining a set of transfer functions that operate over each segment. Modifications of the function samples can be used to interactively modify the segmentation of the data, while interactions with transfer functions can be used to interactively generate meaningful visualizations of the data. These interaction techniques provide users with the ability to quickly and directly resolve collisions created by dimension reduction (i.e., when dissimilar high-dimensional values map to similar low-dimensional values). We exhibit a system where feature segmentation does not rely upon fragile high-dimensional queries or clustering, and within which users have great flexibility in exploring complex function fields.

6.1 Range-Space Segmentation

Consider a function-space distance metric $||\cdot||$ such that $||f-g||$ represents the "similarity" of function f to g. Using such a metric it is possible to project a function field to a scalar field by comparing each of the dataset's functions against a known, exemplar function f. The scalar value at point p with corresponding function f_p is defined to be the distance in function-space between f_p and the exemplar function:

$$S^f : p \in \mathbb{R}^n \rightarrow ||f - f_p||. \tag{6.1}$$

We can extend this approach to produce a *range-space segmentation* of a function field. Consider an ordered set of m function samples $\mathbf{M} = (f_1, f_2, \ldots, f_m)$. From such a set, we can construct *multiple* scalar fields S^{f_i}, one for each function in $\mathbf{M}$. These fields describe the function space distance from the function at p to each of the functions in $\mathbf{M}$. Range-space segmentations are formed by keeping two pieces of information per point p: first, a *classification field* value with the index i of the function f_i in $\mathbf{M}$ that is closest to f_p, and second, a *multi-function scalar distance field* value that stores the distance from f_i to f_p.

Thus, in the multi-function scalar distance field S^*, the scalar value at point p becomes the minimal function-space distance from f_p to any function in $\mathbf{M}$:

$$S^* : p \in \mathbb{R}^n \rightarrow \min_{i \in (1,\ldots,m)} ||f_i - f_p||. \tag{6.2}$$

S^* can be calculated from a set of function samples $\mathbf{M}$ either by computing each scalar distance field S^{f_i} and then their minimum, or by computing the minimum for each point p sequentially.

We also generate an integer-valued classification field L that specifies the *index* of the function in $\mathbf{M}$ used to minimize the value at point p in S^* (rather than the minimum value itself):

$$L : p \in \mathbb{R}^n \rightarrow \arg\min_{i \in (1,\ldots,m)} ||f_i - f_p||. \tag{6.3}$$

For example, if the nth sample is used to minimize Equation (6.2) at point p, then $L(p) = n$. Under

a Euclidean distance metric, one way to interpret the value at a point p in L is as the label of the cell to which f_p belongs in the function-space Voronoi tessellation [6] created by the function samples.

These two scalar fields, S^* and L, represent the range-space segmentation of the function field formed by the set of function samples $\mathbf{M}$. Before turning to direct visualization and feature segmentation, however, we must discuss two important aspects of the segmentation construction process: what similarity metric to use, and how to choose function samples.

6.1.1 Similarity Measures

Our approach is very flexible with respect to the distance metric used to create the range-space segmentation. To obtain results, the distance metric simply needs to reflect a measure of similarity between two function-space samples.

An example of a general, function-space metric is the weighted Euclidean metric. Given a function f defined over a closed interval I, the weighted Euclidean metric is defined as:

$$||f|| = \left(\int_I w(x) f(x)^2 dx \right)^{\frac{1}{2}}, \tag{6.4}$$

where $w(x)$ is a weight function. If f is defined discretely – i.e., represented by a sequence of m points $(f_1, f_2, ..., f_m)$, the metric is defined as:

$$||f|| = \left(\sum_{i=1}^{m} w_i f_i^2 \right)^{\frac{1}{2}}, \tag{6.5}$$

where $(w_1, w_2, ..., w_m)$ are a set of weights. We measure the distance between two functions f and g by calculating $||f - g||$. An interesting note is that if the weight function (or vector) is constant, then the segmentation produced by multiple function-space samples under this metric corresponds to a Voronoi tessellation [6] of the range-space.

We have applied the above metric over hyperspectral imagery and particulate pollution data with good results (Section 6.2). However, in these and other application domains the choice of dis-

tance metric will lead to different range-space segmentation results. In addition to weighted Euclidean, other commonly used metrics in the context of sampled functions include Earth Mover's Distance [65], and Chang's spectral distance metrics [13]. Cox and Cox [18] also suggest a number of additional metrics.

6.1.2 Function Samples

In order to create a range-space segmentation, the user must specify a set of function samples **M**. In some situations, users might have meaningful exemplar functions *a priori* in the form of "test sets." An example is in the domain of hyperspectral imagery, where it is likely that analysts already have a list of reflectance functions corresponding to known materials (i.e., a spectral library). Other domains are also likely to have their own "known" function signatures, and our approach fully supports this type of foreknowledge.

In our software implementation, we provide the user with flexible controls to specify the function samples, including:

- functions from test sets,
- analytic functions,
- hand-drawn functions, and
- functions derived from the data under various distribution statistics.

In addition, we allow users to specify function samples through an interactive spatial probing process. A *probe* is a user-specified point in the data domain $p \in \mathbb{R}^n$, controlled by a *full space cursor* [56]. The function sample associated with the probe is the function f_p at the point p in the function field.

6.1.3 Visualization

Range-space segmentations are effective vehicles for producing direct visualizations of function field data. A segmentation is the combination of a distance field and an integer-valued classification field, both of which are scalar fields in $\mathbb{R}^n$. We can apply traditional scalar field rendering techniques, largely unchanged, upon a range-space segmentation to produce images of two-dimensional function fields and volume renderings of three-dimensional fields.

In the case where the segmentation is created by a single function sample – i.e., $m = 1$, the user can directly associate colors with scalar values in the distance field. For volume rendering, where a color's opacity is important, users are able to modify an opacity function as part of the transfer function. We use 1D transfer functions during volume rendering, but two-dimensional [47], multi-dimensional [44], and local [50] transfer functions may be applied to emphasize local structures.

Most often, however, the range-space segmentation will be derived from multiple function samples. To visualize non-trivial segmentations we turn to the classification field. We associate a transfer function with each of the m function samples to create an ordered list of transfer functions $\mathbf{T} = (t_1, \ldots, t_m)$. During rendering, the classification field value at p determines the transfer function $t_{L(p)}$ used to color the scalar value at p in S^*:

$$\text{color}(p) = t_{L(p)}(S^*(p)).$$

Geometrically, we associate a different transfer function within each "cell" defined by the range-space segmentation. Figure 6.1 illustrates this rendering approach, in which a point p is shaded using the transfer function associated with the nearest function sample to f_p.

6.1.4 Feature Segmentation

Range-space segmentations facilitate the construction of segmenting surfaces between feature regions in two and three dimensions. The key to producing segmenting surfaces in our framework is to perform surface extraction over the classification field L. Recall that integer values in the clas-

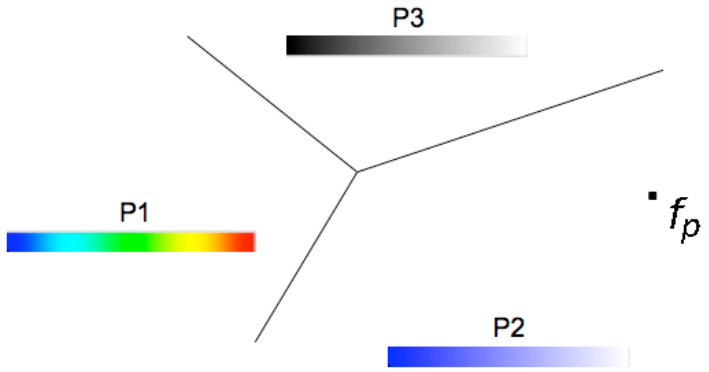


Figure 6.1: Our approach can be interpreted as creating a Voronoi-like tessellation of the function field range space (i.e., space of functions). Each "cell" of the tessellation is assigned its own colormap for visualization. The color of a point p is determined by the location of its corresponding function f_p within the range-space segmentation.

sification field encode the range-space segmentation "cell" membership for each point, as defined by the similarity metric and the current function samples (Equation 6.3). Thus, surfaces that partition the classification field into homogeneously labeled regions correspond to boundaries between function-space features.

For a range-space segmentation constructed from two function samples, the classification field will be a binary labeling of the function field domain. We can extract the segmenting surface(s) between features by performing isosurfacing with an isovalue between the two labeling values (e.g., isovalue of 0.5 when the labels are 0 or 1). Algorithms such as Marching Cubes [48] can be used to extract a surface representing the set of points I with a constant isovalue v through a the classification field – i.e., $I : \{x|L(x) = v\}$.

In more complex cases, where the classification represents a segmentation derived from three or more function samples, segmenting surfaces can be extracted using one of various multi-label segmentation algorithms. Examples include multi-label Marching Cubes methods [31, 84, 7], Dual Contouring [39], or the method of Nielson and Franke [57] on an implicit tetrahedrization of the rectilinear domain.

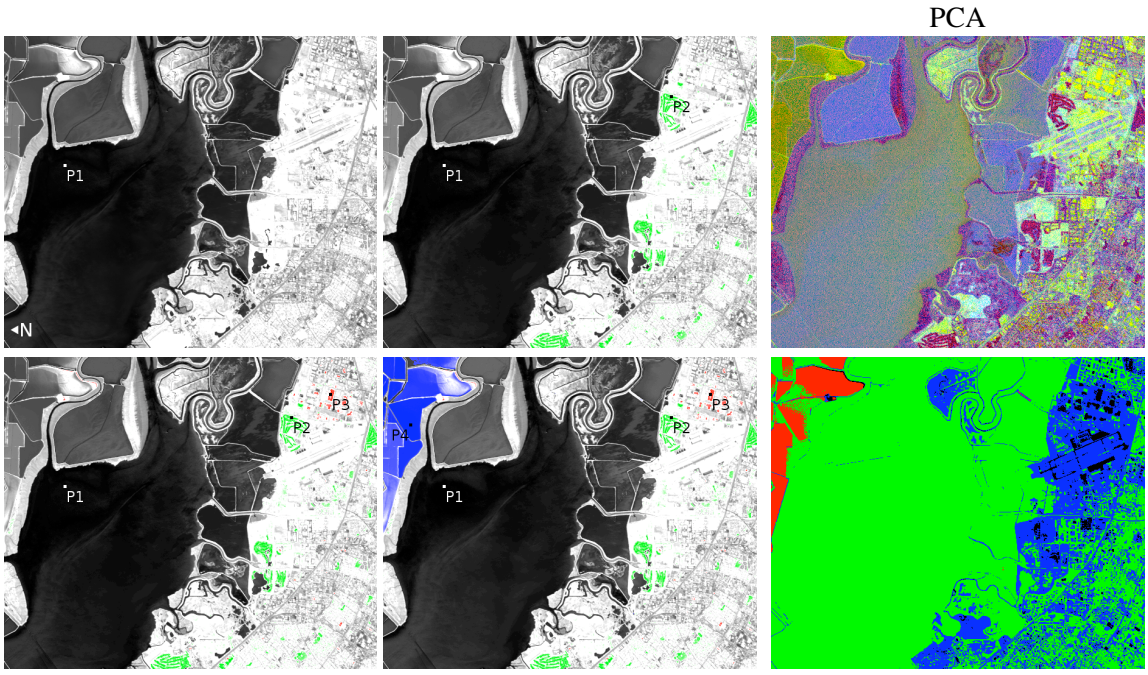


Figure 6.2: Visualizations of a hyperspectral image of Moffett Field and the San Francisco Bay. The leftmost set of images shows the construction of a range-space segmentation with 1, 2, 3, and 4 probes. On the right are images generated by mapping PCA components to RGB (top), and by Vector Quantization (VQ) clustering (bottom).

6.2 Results

In this section we present results on multiple function field datasets, produced by remote sensing instruments and atmospheric pollution simulation. These application domains were introduced in Chapter 5.

6.2.1 Hyperspectral Imagery

The Airborne Visible InfraRed Imaging Spectrometer (AVIRIS) [80] is an aircraft-mounted hyperspectral imagery acquisition system that produces calibrated 614x512 images of up-welling spectral radiance. In AVIRIS images each pixel consists of 224 radiance (or reflectance) samples over visible and short-wave infrared wavelengths, yielding an image size of approximately 270 megabytes.

The leftmost images of Figure 6.2 show the incremental construction of a range-space segmenta-

tion for a hyperspectral image of Moffett Field and the San Francisco Bay. The probes, and their associated transfer functions, were interactively added in the following order: 1) over water with function (black-to-white), 2) on a golf course with function (green), 3) on a building with function (red), and 4) over evaporation ponds containing brine shrimp with function (blue). Because segmentation is done in the original spatial and functional domains, users can track particular features of interest while still "managing" unknown features with tentative function samples and transfer functions. It is often the case that the context provided by the initial distance field visualization helps the user identify and segment addition features. Furthermore, spatial coherency in the function field helps our visualizations to remain relatively stable when adding and changing function samples by probing.

In the upper right image of Figure 6.2 we show the result of applying PCA over the hyperspectral image for visualization [79, 34]. Here, individual dimensions are mapped to RGB color channels after the PCA transform; we have used the mapping $(P_4, P_5, P_6) \rightarrow (R, G, B)$. Unlike our method, PCA requires an expensive preprocess of the data and is a "static" dimension reduction. The only choice in PCA visualization is the set of principle components to consider. This can be difficult: we found (P_4, P_5, P_6) to be the first set of components that produce an image not dominated by noise, and cycling through additional sets of components leads to little additional insight into the data.

We also compare our approach to clustering. The bottom right image of Figure 6.2 shows the results of applying Vector Quantization (VQ) clustering [2]. We have clustered the first 10 components from a PCA transform of the hyperspectral image into four clusters; clustering over all 224 dimensions in either the original or PCA transformed data produces an *extremely* noisy clustering. With the correct settings and preprocessing, clustering is able to capture similar features to our method (e.g., the brine shrimp ponds in the image), because both are based upon a segmentation of function range-space.

6.2.2 Particulate Pollution

We now consider two time-varying, three-dimensional particulate pollution datasets produced from air quality simulations. The first dataset (National) is a 148x112x19 grid of particulate H_2O concentration over the continental United States. Our second dataset (CRPAQS), from the California Regional Particulate Air Quality Study, is a 185x185x15 grid of particulate SO_4 concentration throughout the San Joaquin Valley, California, U.S.A. Each dataset contains cell-centered, 9-sampled functions of particle concentration versus diameter, over 25 timesteps. The CRPAQS dataset is approximately 450 megabytes, and the National dataset is approximately 260 megabytes (much larger than scalar fields with similar spatial extents).

Important to our method is that by using multiple probes, and simple transfer functions, users are able to create renderings that meaningfully highlight different aspects of the same dataset. Figure 6.3 shows H_2O concentration from the National particulate dataset rendered over multiple timesteps using two probes. In both (a) and (b), the first probe, with a black-to-white transfer function, is located over central Mexico, and corresponds to low H_2O concentration. In (a), the second probe, with a rainbow transfer function, is located in an area of low to moderate moisture in the United States mid-west. In (b), the second probes is placed in an localized area of functions with high total moisture content.

Our method also provides flexibility in the visualization and segmentation of time-varying function fields. In time-varying fields, probes become points in $\mathbb{R}^n \times \mathbb{T}$. In Figure 6.4(a), we show a direct visualization of the range-space segmentation created by three function samples in different timesteps for the CRPAQS dataset. The first probe is located outside of the central San Joaquin valley and has low total SO_4 concentration. The second and third probes, however, are located at the same spatial position, but at different points in time. The second probe with a red transfer function is in timestep 0 and corresponds to function of high total SO_4 concentration. The third probe with a blue transfer function is in timestep 18 and corresponds to moderate SO_4 concentration. Figure 6.4(b) shows a closeup of the direct visualization produced by the range-space segmentation. In 6.4(c) we highlight feature segmentation: multi-material surface extraction as described in Section 6.1.4

Table 6.1: Timings for range-space segmentation creation.

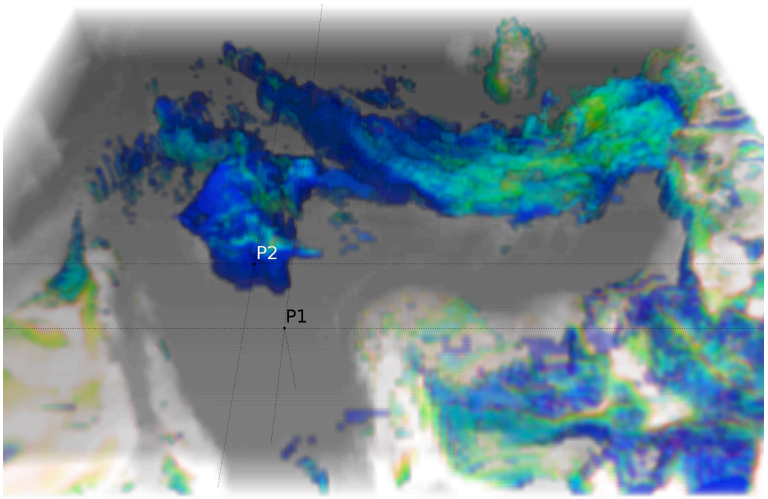
Dataset	S	S^*	**Total** (ms)
Hyperspectral Imagery	80	16	336
H_2O Aerosol (National)	20	11	51
SO_4 Aerosol (CRPAQS)	27	23	104

is used to extract boundaries between spatial regions with functions having high, medium, and low total SO_4 concentration.

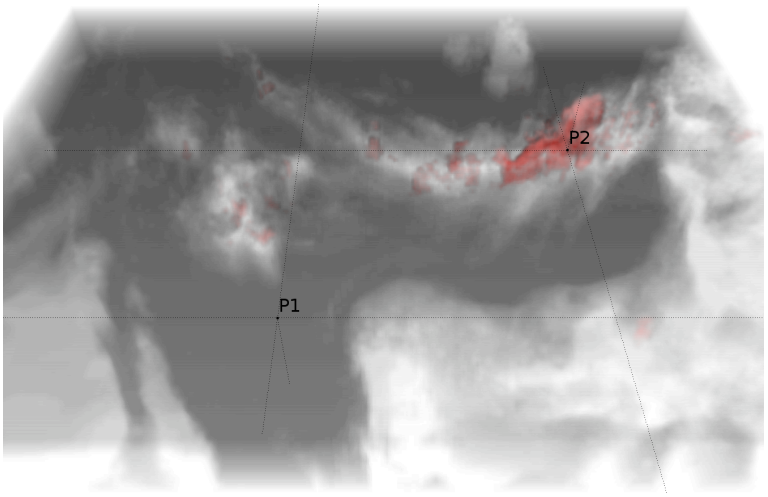
6.2.3 Performance

The techniques presented herein are best utilized in an interactive setting, where operations such as changing function samples, creating segmentations, and deriving new visualizations are rapidly realized. We have performed testing on an Apple MacBook Pro notebook computer (2.33 GHz Intel Core 2 Duo processor, 2 GB memory, and an ATI Radeon X1600 graphics card). For the datasets considered our method is interactive.

Table 6.1 shows timings in milliseconds for the generation of the segmentations used in Figures 6.2, 6.3, and 6.4(b) (four, two, and three probes, respectively). The column S lists the time required to generate one single-function distance field (Equation 6.1). The S^* column lists the time required to combine all distance fields into a range-space segmentation (Equations 6.2 and 6.3). The totals listed reflect the time required to *fully* generate a new segmentation: i.e., to generate each single-probe field and combine them into a multi-probe field. Often, however, end users will experience far less latency. When the user modifies a function sample (for example, by repositioning a probe), they only modify one of the m distance fields, which the remaining $m-1$ fields remain unchanged. Thus, the latency experience by users when changing a single function sample will be $S+S^*$ from Table 6.1.

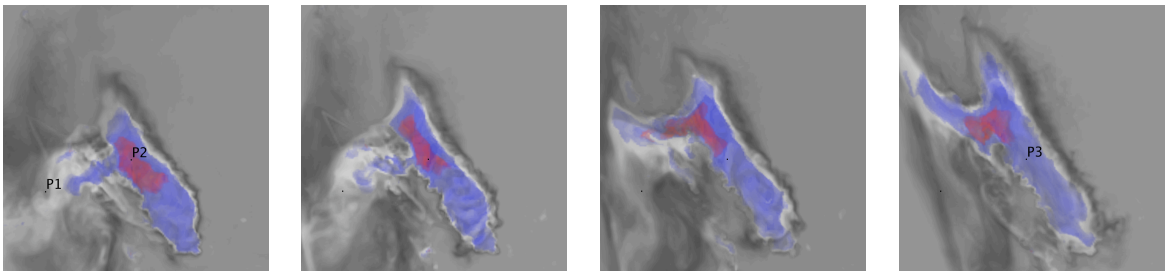


(a) Broad region of moderate moisture functions

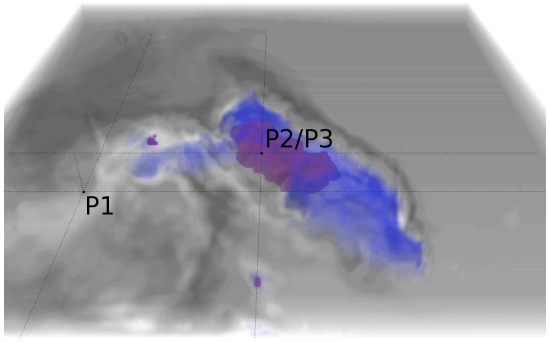


(b) Localized region of high moisture functions

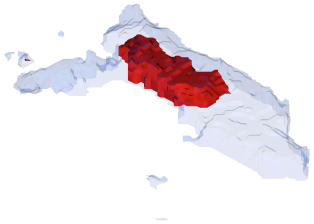
Figure 6.3: Volume renderings produced from range-space segmentations of the National H_2O particulate concentration dataset. By using multiple probes, and simple transfer functions, users are able to create renderings that meaningfully highlight different aspects of the same dataset.



(a) Time Probes



(b) Volume Rendering



(c) Feature Segmentation

Figure 6.4: Range-space segmentation of the CRPAQS dataset using multiple probes in different timesteps: the first two probes are in timestep 0, while the third probe is in the last timestep. In (a) we use these probes to visualize the movement of high (red) and moderate (blue) SO_4 concentration features over time through the San Joaquin Valley. In (b), we show a closeup of the first timestep, while (c) shows the segmentation of high, moderate, and low concentration regions.

Chapter 7

Feature Extraction with Queries

In addition to visual exploration, the extraction of features is important for quantitative analysis and annotation. Combining visual exploration with feature extraction opens the door to performing quantitative analysis, such as calculating the size of a body of water or determining how long a pollution source remains active. It also makes it simple to annotate function fields with overlays. In this chapter we explore the extraction of function field features with queries.

Query-Driven Visualization (QDV) techniques integrate database technologies and visualization strategies to address the continually increasing size and complexity of scientific datasets: large data is intelligently pared down by user-specified "selection" queries, allowing smaller, more meaningful subsets of data to be efficiently analyzed and visualized. Well-characterized range queries are capable of identifying spatial regions where many domain-specific events occur: combustion flame fronts, vortices, chemical reaction fronts, etc.

Stockinger et al. [75] were first to present the notion of coupling visualization with high performance query technology. Their work demonstrates that the computational complexity of visualization processing can be constrained to the number of items returned by a query. Their approach introduces a software system (DEX), that utilizes a highly efficient indexing and query infrastructure, called FastBit, to rapidly identify and visualize "regions of interest" within a dataset [29, 74, 75, 76].

Specified as Boolean range queries, these regions of interest tend to be significantly smaller subsets of the original dataset; thus, these regions require less time and computational effort to analyze, visualize, and interpret.

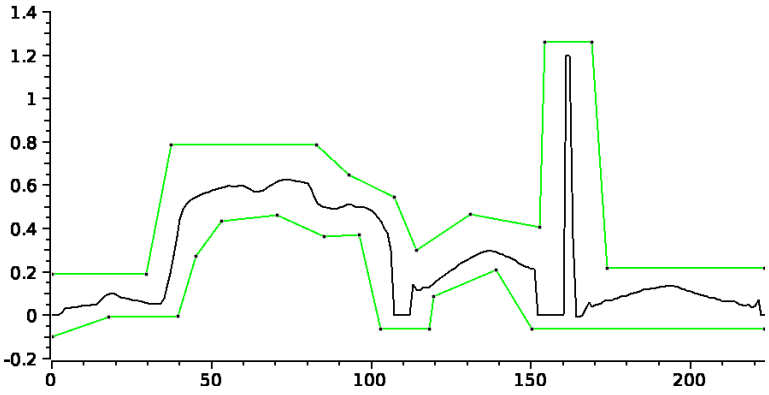
In this chapter, we apply QDV techniques to extract function field features. We consider features in function fields to be spatial regions in which the 1-dimensional functions are similar. In the next section we define feature queries and describe their evaluation over a function fields; we demonstrate a number of queries to extract features such as golf courses, water, and areas of high pollution, and show that query constraints are reusable across multiple datasets. In Section 7.2 we evaluate the performance of function field queries.

7.1 Feature Extraction

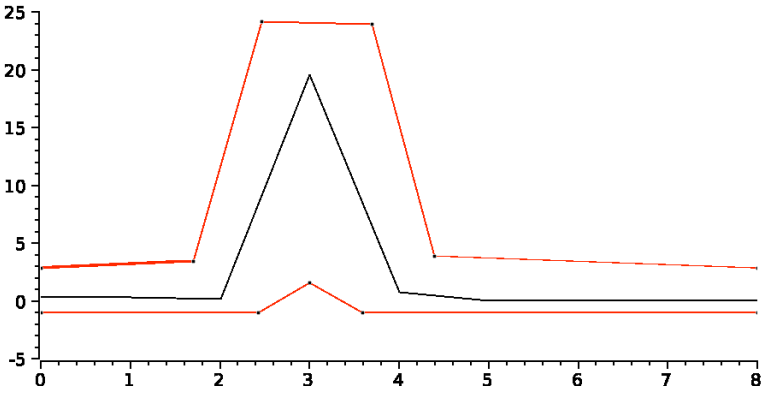
We define a feature query as a set of constraints over the closed interval I. For a dataset with m samples per function, these constraints take the form of minimum-maximum intervals Q_i for each sample, $i = 1, \ldots, m$. A point p in the dataset with 1-dimensional function f_p, is part of the feature if and only if $f_{p_i} \in Q_i$ for $i = 1, \ldots, m$.

In our system, users first explore function fields using distance field renderings. Once a feature has been identified, the user is able to sketch a pair of curves that define the feature query's minimum and maximum constraints. For example, Figure 7.1(a) shows the constraint curves, in green, used to extract golf courses from hyperspectral images; the black curve plots the function of a pixel from a golf course. Figure 7.2 shows two hyperspectral images annotated with overlays produced by three queries: golf courses (green), water (blue), and evaporation ponds containing brine shrimp (red).

Defining features as a set of constraints in function-space makes the query constraints reusable across multiple datasets. The queries used to extract features in Figure 7.2 were constructed by a user exploring the hyperspectral image of Moffett Field and the San Francisco Bay in 7.2(a). The image in 7.2(b) shows an area approximately 18 kilometers to the east of Moffett Field; golf courses and water were extracted using the pre-constructed queries without modification.

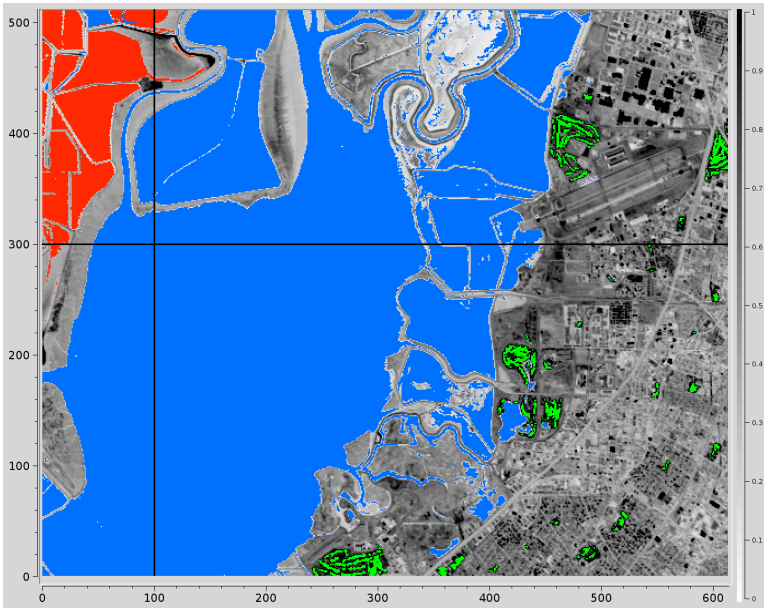


(a) Golf Courses

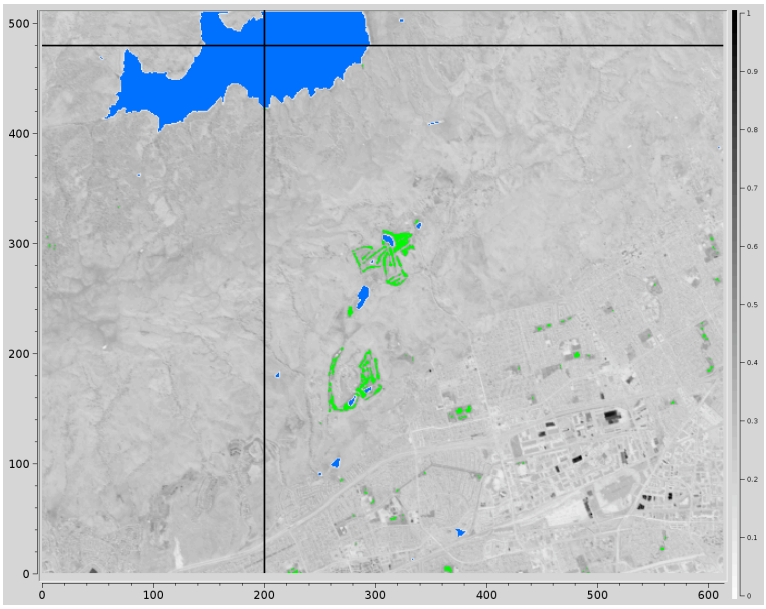


(b) High SO_4 Concentration

Figure 7.1: In (a), we show the function-space constraints (green) used to extract golf courses from hyperspectral images (Figure 7.2). In (b), we show the constraints (red) used to extract regions from the CRPAQS dataset in which medium-sized SO_4 particles have high concentration (Figure 7.3). In both, the black curves are functions that satisfy the feature queries.



(a)



(b)

Figure 7.2: Hyperspectral images annotated with overlays produced by three queries: golf courses as shown in Figure 7.1(a) (green), water (blue), and evaporation ponds containing brine shrimp (red). The feature queries were constructed by a user exploring the hyperspectral image of Moffett Field and the San Francisco Bay in (a). The image in (b) shows an area approximately 18 kilometers to the east of Moffett Field; golf courses and water were extracted using the pre-constructed queries without modification.

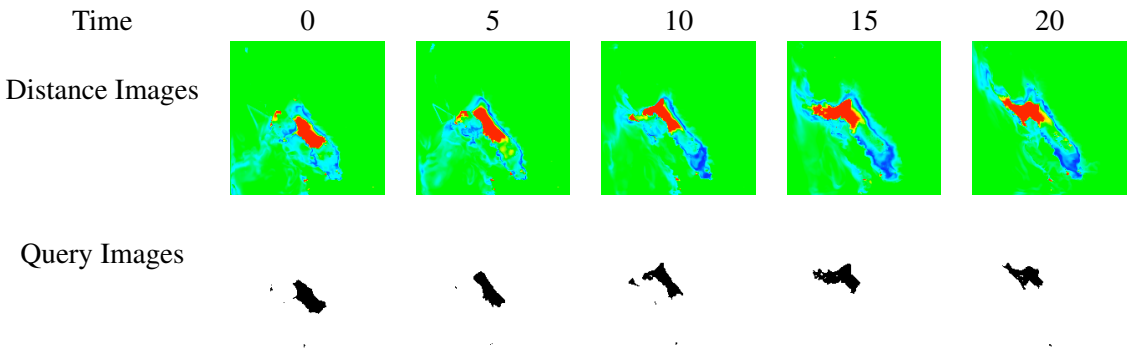


Figure 7.3: Distance field renderings generated from the CRPAQS dataset, and the results of using the feature query shown in Figure 7.1(b) to extract regions in which medium-sized SO_4 particles have high concentration. For clarity we only show ground layer images from the three-dimensional results.

Feature queries work on datasets of arbitrary spatial dimension, and upon time-varying datasets. Figure 7.1(b) shows a simple query that can be used to extract regions from the CRPAQS dataset in which medium-sized SO_4 particles have high concentration. In time-varying datasets it is often possible to reuse a query across multiple timesteps. Figure 7.3 shows distance field renderings and the regions extracted by the aforementioned query for timesteps 0, 5, 10, 15, and 20. For clarity we only show ground layer images from the three-dimensional results.

7.2 Implementation & Performance

The datasets used in this chapter were previously described in Section 5.2. The first dataset contains multiple AVIRIS hyperspectral images of Moffett Field and the San Francisco Bay area. The second function field dataset is an air quality simulation from the California Regional Particulate Air Quality Study (CRPAQS).

We have tested our methods on a 2.6 Ghz Mobile Pentium 4-M laptop with 1.0 Gb RAM and a nVidia GeForce 4200 Go graphics card. Figure 7.4 shows part of our software system. The upper plot shows the probe function in black, and the constraint curves in red defining the query that extracts evaporation ponds containing brine shrimp from hyperspectral images. The lower plot shows the sample weights curve. The minimum-maximum constraint curves and sample weights

Dataset	Query	Execution (ms)	Coverage
Hyperspectral Image	Water	320	46.11
	Golf Courses	218	0.82
	Evaporation Ponds	257	3.66
CRPAQS	High SO_4 Concentration	23	0.85

Table 7.1: Timings and coverage. For hyperspectral images, distance fields are generated at the rate of approximately 6 per second; for the CRPAQS dataset, approximately 25 per second. Regions extracted by the feature queries are shown in Figures 7.2(a) and 7.3.

curve are modifiable by the user; control points can be added, removed, and manipulated. The right side of the interface provides more controls for feature queries.

Feature queries evaluate rapidly in our system, thus allowing users to interactively change function-space constraints. Table 7.1 shows timing results and coverage for query evaluation. Single Instruction, Multiple Data (SIMD) instructions (e.g., SSE2 for Intel processors) have been used to vectorize the code for query evaluation. In all function field datasets, multiple queries may be evaluated per second. Coverage, the percentage of total cells returned by a query, is an example of quantitative analysis facilitated by feature extraction.

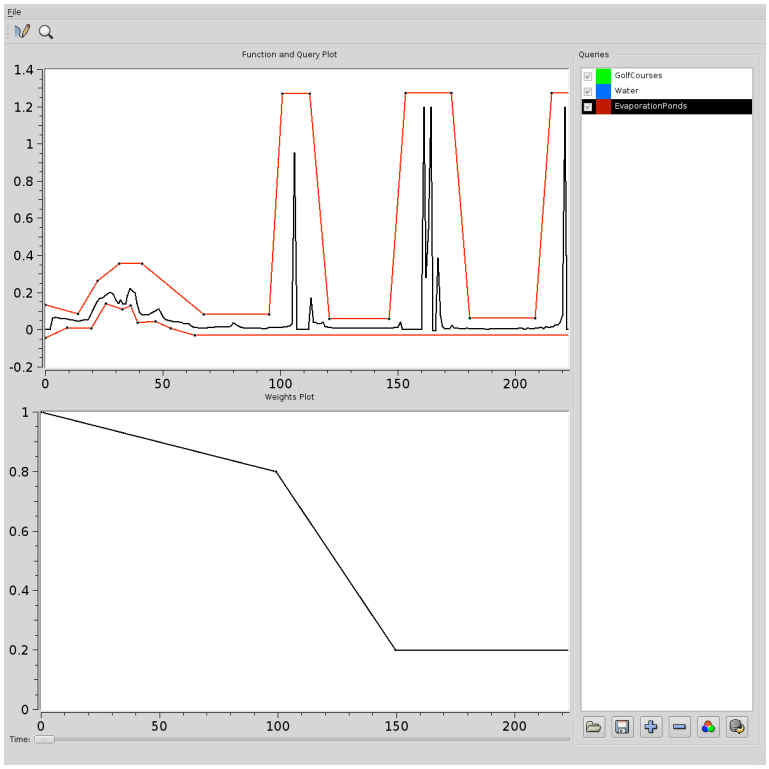


Figure 7.4: Part of our software system. The upper plot shows the probe function in black, and the constraint curves in red defining the query that extracts evaporation ponds containing brine shrimp from hyperspectral images. The lower plot shows the sample weights curve. The minimum-maximum constraint curves and sample weights curve are modifiable by the user; control points can be added, removed, and manipulated. The right side of the interface provides more controls for feature queries.

Part III

Summary

Chapter 8

Conclusion and Future Work

This dissertation has presented novel research on two fronts: material interface reconstruction for volume fraction data, and visualization and analysis methods for function fields.

Material Interface Reconstruction

In Chapter 3 we presented a discrete approach to MIR based upon optimizing the labeling of fractional volume elements over a subdivided spatial domain. We introduced a *volume conservative swap* move for the optimization process, and discussed methods for extracting and visualizing material interfaces from a labeled discretization. Our technique gives significantly better results than previous methods, producing interfaces between multiple materials that are continuous across cell boundaries for time-varying and static data in arbitrary dimension with bounded error.

Chapter 4 extended these results; first, an interface topology generation pipeline was developed, and second iterative surface improvement was accomplished using a volume-adaptive active interface model. This new material interface reconstruction method produces high-quality boundary meshes with low error in two and three dimensions. Experimental results show our approach to be very well-behaved: per-cell error tends to be significantly less than 1%, while producing continuous,

piecewise linear meshes.

The primary software artifact of the material interface reconstruction research presented in this dissertation is a cross-platform research software called *Gallimaufry* (a confused jumble or medley of things). This software performs discrete and volume-adaptive reconstructions of arbitrarily many materials over 2D and 3D rectilinear grids. Gallimaufry also provides a simple framework for implementing existing reconstruction algorithms (SLIC [59] and PLIC [86], currently) and testing new reconstruction ideas. Reconstructions can be visualized *in situ*, or output in multiple image and surface mesh formats. We have also implemented our discrete reconstruction method in a version controlled "branch" – *jcanders/dev* – of the scientific visualization software VisIt [15] from LLNL (`http://www.llnl.gov/visit`).

There remains, however, future work in terms of algorithms and software:

- Develop methods to generate a parameterized range of topology solutions. For example, users should be able to control – on a per-material basis – whether they want interfaces that tend toward thin filaments/shells or multiple disconnected blobs.
- Explore the use of level set methods [61] as an alternative to our current mesh-based active interface model, especially in the presence of parameterized topology control.
- Support unstructured meshes and adaptive mesh refinement (AMR) grids. In our proposed framework, this would require the extension of our topology generation pipeline to support different mesh types; the volume-adaptive active interface model presented is largely independent of the underlying mesh.
- It is not uncommon for volume fraction datasets to contain billions of elements; additional work should focus upon the speed of our algorithm through parallelism, simpler surface meshes, and possibly multi-resolution methods.
- Volume fraction data is often dumped on a coarse timescale from fine-grained simulations. Interface tracking at the visualization time scale should be investigated to encourage timestep-to-timestep topology consistency when possible.

Other domains, besides simulation, are likely to benefit from robust, volume conservative interface reconstruction techniques. A concrete example is spectral imagery, introduced in Chapter 5, where each pixel is a sampled function of radiance versus wavelength, rather than a tristimulus RGB value. From spectral functions it is possible to estimate the ratios of different materials present at each pixel [71]. We look forward to applying our methods in this setting to reconstruct sub-pixel interfaces within spectral images, as well as to other problem with volume fraction data.

Function Fields

Chapters 6 and 7 presented a range-space segmentation framework for the visualization and analysis of function field data: one of the myriad of possible data types that can populate the variables in a multi-dimensional, multi-variate dataset. The presented methods increase our capacity to visualize these complex fields, and help us gain new insight about the data. These methods are interactive, and have been useful for exploring, annotating, and performing quantitative analysis on function fields from multiple application domains.

Areas of future work should include optimizations to ensure interactivity when working with very large function fields, as well experimentation with different feature criterion. Future work to generalize and extend the presented approaches to other types of multi-variate data is also warranted.

Bibliography

[1] D. Adalsteinsson and J.A. Sethian. A fast level set method for propagating interfaces. *J. of Computational Physics*, 118(2):269–277, 1995.

[2] Stanley C. Ahalt, Ashok K. Krishnamurthy, Prakoon Chen, and Douglas E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3(3):277–290, 1990.

[3] Jörgen Ahlberg. Active contours in three dimensions. Master's thesis, Linköping University, September 1996. LiTH-ISY-EX-1708.

[4] Bruno Aiazzi, Stefano Baronti, Leonardo Santurri, Massimo Selva, and Luciano Alparone. Information-theoretic assessment of multi-dimensional signals. *Signal Process.*, 85(5):903–916, 2005.

[5] John C. Anderson, Christoph Garth, Mark A. Duchaineau, and Kenneth I. Joy. Discrete multi-material interface reconstruction for volume fraction data. *Computer Graphics Forum*, 27(3):1015–1022, May 2008.

[6] Franz Aurenhammer. Voronoi diagrams – survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[7] David C. Banks and Stephen Linton. Counting cases in Marching Cubes: Toward a generic algorithm for producing substitopes. In *Proc. of IEEE Visualization*, pages 51–58, October 2003.

[8] David J. Benson. Computational methods in lagrangian and eulerian hydrocodes. *Comput. Methods Appl. Mech. Eng.*, 99(2-3):235–394, 1992.

[9] Kathleen S. Bonnell, Mark A. Duchaineau, Daniel R. Schikore, Bernd Hamann, and Kenneth I. Joy. Material interface reconstruction. *IEEE Trans. on Visualization and Computer Graphics*, 9(4):500–511, 2003.

[10] Kathleen S. Bonnell, Daniel A. Schikore, Mark A. Duchaineau, Bernd Hamann, and Kenneth I. Joy. Constructing material interfaces from data sets with volume-fraction information. In *Proc. of IEEE Visualization*, pages 367–372, October 2000.

[11] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[12] Ken Brodlie and Jason Wood. Recent advances in volume visualization. *Computer Graphics Forum*, 20(2):125–148, 2001.

[13] Chein-I Chang. An information-theoretic approach to spectral variability, similarity, and discrimination for hyperspectral image analysis. *IEEE Transactions on Information Theory*, 46(5):1927–1932, 2000.

[14] R. K. Cheng. Low swirl combustion. In *Gas Turbine Handbook*. U.S. Department of Energy, 2006.

[15] Hank Childs, Eric S. Brugger, Kathleen S. Bonnell, Jeremy S. Meredith, Mark Miller, Brad J. Whitlock, and Nelson Max. A contract-based system for large data visualization. In *Proc. of IEEE Visualization*, pages 190–198, 2005.

[16] L. D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing*, 53(2):211–218, 1991.

[17] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.

[18] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling*. Chapman & Hall/CRC, second edition, September 2000.

[19] Scott Dillard, John Bingert, Dan Thoma, and Bernd Hamann. Construction of simplified boundary surfaces from serial-sectioned metal micrographs. *Trans. on Visualization and Computer Graphics*, 13(6):1528–1535, 2007.

[20] V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction. Technical Report LA-UR-05-7571, Los Alamos National Laboratory, October 2005.

[21] V. Dyadechko and M. Shashkov. Multi-material interface reconstruction from the moment data. Technical Report LA-UR-06-5846, Los Alamos National Laboratory, December 2006.

[22] V. Dyadechko and M. Shashkov. Reconstruction of multi-material interfaces from moment data. *Journal of Computational Physics*, 2008. To appear.

[23] Charles R. Ehlschlaeger, Ashton M. Shortridge, and Michael F. Goodchild. Visualizing spatial data uncertainty using animation. *Computational Geosciences*, 23(4):387–395, 1997.

[24] Zhe Fang, Torsten Möller, Ghassan Hamarneh, and Anna Celler. Visualization and exploration of time-varying medical image data sets. In *GI '07: Proceedings of Graphics Interface 2007*, pages 281–288, New York, NY, USA, 2007. ACM Press.

[25] David A. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4(6):709–712, 1988.

[26] R. V. Garimella. Mesh data structure selection for mesh generation and FEA applications. *International J. for Numerical Methods in Engineering*, 55(4):451–478, 2002.

[27] R. V. Garimella, V. Dyadechko, B. K. Swartz, and M. J. Shashkov. Interface reconstruction in multi-fluid, multi-phase flow simulations. In *Proc. of International Meshing Roundtable*, pages 19–32, September 2005.

[28] Sarah F. F. Gibson. Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. *Lecture Notes in Computer Science*, 1496:888–900, 1998.

[29] Luke Gosink, John Shalf, Kurt Stockinger, Kesheng Wu, and Wes Bethel. Hdf5-fastquery: Accelerating complex queries on hdf datasets using fast bitmap indices. In *SSDBM*, pages 149–158, 2006.

[30] Fran çois Graner and James A. Glazier. Simulation of biological cell sorting using a two-dimensional extended Potts model. *Physical Review Letters*, 69(13):2013–2016, September 1992.

[31] Hans-Christian Hege, Martin Seebaß, Detlev Stalling, and Malte Zöckler. A generalized Marching Cubes algorithm based on non-binary classifications. ZIB Preprint SC-97-05, 1997.

[32] William L. Hibbard, John Anderson, Ian Foster, Brian E. Paul, Robert Jacob, Chad Schafer, and Mary K. Tyree. Exploring coupled atmosphere-ocean models using Vis5D. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(2/3):211–222, Summer/Fall 1996.

[33] C.W. Hirt and B.D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Computational Physics*, 39:201–225, 1981.

[34] N. Jacobson and M. Gupta. Design goals and solutions for display of hyperspectral images. In *IEEE Image Processing*, volume 2, pages 622–625, September 2005.

[35] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[36] Jr. James Edward Pilliod and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *J. Computational Physics*, 199(2):465–502, 2004.

[37] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.

[38] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[39] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of Hermite data. *ACM Trans. on Graphics*, 21(3):339–346, 2002.

[40] D. Kao, M. Kramer, A. Love, J. Dungan, and A. Pang. Visualizing distributions from multi-return lidar data to understand forest structure. *Cartographic Journal, Special Issue on Geo-Visualization*, 42(1):1–14, June 2005.

[41] David Kao, Alison Luo, Jennifer L. Dungan, and Alex Pang. Visualizing spatially varying distribution data. *Information Visualization*, pages 219–226, 2002.

[42] M. Kass, A. Witkin, and D. Terzopoulos. Snakes – active contour models. *International J. of Computer Vision*, 1(4):321–331, 1987.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598(4598):671–680, May 1983.

[44] Joe Kniss, Gordon Kindlmann, and Charles Hansen. *Multidimensional Transfer Functions for Volume Rendering*, chapter 9, pages 189–210. Elsevier, 2005.

[45] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? In *Proc. European Conference on Computer Vision–Part III*, pages 65–81, 2002.

[46] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, July 2004.

[47] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 08(3):29–37, 1988.

[48] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Proc. of SIGGRAPH*, pages 163–169, 1987.

[49] Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. Multiple interacting liquids. *ACM Trans. on Graphics*, 25(3):812–819, 2006.

[50] Claes Lundstrom, Patric Ljung, and Anders Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1570–1579, 2006.

[51] Alison Luo, David Kao, Jennifer Dungan, and Alex Pang. Visualizing spatial distribution data sets. In C. D. Hansen G.-P. Bonneau, S. Hahmann, editor, *Proceedings of the Symposium on Data Visualisation*, pages 29–38, Grenoble, France, 2003. Eurographics Association.

[52] J. MacQueen. Some methods for classification and analysis of multivariate observations. pages 281–297, 1967.

[53] Tim McInerney and Demetri Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1:91–108, June 1996.

[54] J. S. Meredith. Material interface reconstruction in VisIt. In *NECDC*, October 2004.

[55] Barry Merriman, James K. Bence, and Stanley J. Osher. Motion of multiple junctions: a level set approach. *J. Computational Physics*, 112(2):334–363, 1994.

[56] Gregory M. Nielson and Jr. Dan R. Olsen. Direct manipulation techniques for 3d objects using 2d locator devices. In *SI3D '86: Proceedings of the 1986 Workshop on Interactive 3D Graphics*, pages 175–182, New York, NY, USA, 1987. ACM Press.

[57] Gregory M. Nielson and Richard Franke. Computing the separating surface for segmented data. In *Proc. of IEEE Visualization*, pages 229–233, October 1997.

[58] Gregory M. Nielson and Bernd Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *IEEE Visualization*, pages 83–91, 1991.

[59] W. F. Noh and P. Woodward. SLIC (Simple Line Interface Calculation). In A. I. van de Vooren and P. J. Zandbergen, editors, *LNP Vol. 59: Some Methods of Resolution of Free Surface Problems*, pages 330–340, 1976.

[60] S. Osher and R. Fedkiw. *The Level Set Method and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[61] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. of Computational Physics*, 79:12–49, 1988.

[62] Stanley Osher and Ronald Fedkiw. Level set methods: An overview and some recent results. *J. Computational Physics*, 169:463–502, 2001.

[63] Roberto Pinho, Maria Cristina Ferreira de Oliveira, Rosane Minghim, and Marinho G. Andrade. Voromap: A voronoi-based tool for visual exploration of multi-dimensional data. *iv*, 0:39–44, 2006.

[64] William J. Rider and Douglas B. Kothe. Reconstructing volume tracking. *J. Computational Physics*, 141(2):112–152, 1998.

[65] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, pages 59–66, Washington, DC, USA, 1998. IEEE Computer Society.

[66] Steven J. Ruuth. A diffusion-generated approach to multiphase motion. *J. Computational Physics*, 145(1):166–192, 1998.

[67] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. In *Pacific Graphics 2004*, pages 70–76, 2004.

[68] Samuel P. Schofield, Rao V. Garimella, Marianne M. Francois, and Raphal Loubre. Material order-independent interface reconstruction using power diagrams. *International Journal for Numerical Methods in Fluids*, 56(6):643–659, 2008.

[69] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[70] J. A. Sethian. Evolution, implementation, and application of level set and fast marching methods for advancing fronts. *J. Computational Physics*, 169(2):503–555, 2001.

[71] J.J. Settle and N.A. Drake. Linear mixing and the estimation of ground cover proportions. *International J. of Remote Sensing*, 14(6):1159–1177, 1993.

[72] K. A. Smith, F. J. Souls, and D. L. Chopp. A projection method for motion of triple junctions by level sets. *Interfaces and Free Boundaries*, 4(3):263–276, 2002.

[73] J. M. Sotoca, F. Pla, and J. S. Snchez. Band selection in multispectral images by minimization of dependent information. *IEEE Trans. Systems, Man and Cybernetics*, 37(2):258–267, March 2007.

[74] Kurt Stockinger, E. Wes Bethel, Scott Campbell, Eli Dart, and Kesheng Wu. Imaging and visual analysis - detecting distributed scans using high-performance query-driven visualization. In *Supercomputing*, page 82. ACM Press, 2006.

[75] Kurt Stockinger, John Shalf, E. Wes Bethel, and Kesheng Wu. Dex: Increasing the capability of scientific data analysis pipelines by using efficient bitmap indices to accelerate scientific visualization. In *Scientific and Statistical Database Management*, pages 35–44, 2005.

[76] Kurt Stockinger, John Shalf, Kesheng Wu, and E. Wes Bethel. Query-driven visualization of large data sets. In *IEEE Visualization*, pages 167–174, 2005.

[77] Ikuko Takanashi, Shigeru Muraki, Akio Doi, and Arie Kaufman. 3D active net: 3D volume extraction. *Institute of Image Information and Television Engineers*, 51(12):2097–2106, 1997.

[78] Ikuko Takanashi, Shigeru Muraki, Akio Doi, and Arie Kaufman. Three-dimensional active net for volume extraction. In *Proc. of SPIE*, volume 3298, pages 184–193, 1998.

[79] J. S. Tyo, A. Konsolakis, D. I. Diersen, and R. C. Olsen. Principal-components-based display strategy for spectral imagery. *IEEE Trans. Geoscience and Remote Sensing*, 41(3):708–718, March 2003.

[80] G. Vane, R. Green, T. Chrien, H. Enmark, E. Hansen, and W. Porter. The airborne visible infrared imaging spectrometer. In *Remote Sensing Environment*, volume 44, pages 127–143, 1993.

[81] Luminita A. Vese, Tony, and F. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International J. of Computer Vision*, 50:271–293, 2002.

[82] S. A. Wright, S. J. Plimpton, and T. P. Swiler. Potts-model grain growth simulations: Parallel algorithms and applications. Technical Report SAND–97-1925, Sandia National Laboratories, August 1997.

[83] F. Y. Wu. The Potts model. *Reviews of Modern Physics*, 54(1):235–268, January 1982.

[84] Ziji Wu and Jr. John M. Sullivan. Mutliple material Marching Cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, July 2003.

[85] G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley, second edition, 2000.

[86] D. L. Youngs. *Numerical Methods for Fluid Dynamics*, chapter Time-Dependent Multi-Material Flow with Large Fluid Distortion, pages 273–285. Academic Press, 1982.